LEVERAGING THE CLOUD FOR INTEGRATED NETWORK EXPERIMENTATION

THESIS

Brian A. Beam, Major, USA

AFIT-ENG-14-M-11

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-14-M-11

LEVERAGING THE CLOUD FOR INTEGRATED NETWORK EXPERIMENTATION

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyber Operations

Brian A. Beam, B.S. Economics

Major, USA

March 2014

AFIT-ENG-14-M-11

LEVERAGING THE CLOUD FOR INTEGRATED NETWORK EXPERIMENTATION

Brian A. Beam, B.S. Economics
Major, USA

Approved:

//signed//	12 Mar 2014
Douglas Hodson, PhD (Chairman)	Date
//signed//	12 Mar 2014
Kenneth Hopkinson, PhD (Member)	Date
//signed//	12 Mar 2014
Michael Grimaila, PhD, CISM, CISSP (Member)	Date

AFIT-ENG-14-M-11

Abstract

The goal of this research is to determine the feasibility of performing integrated network experimentation using cloud services. This research uses performance metrics to compare computing architectures constructed in the cloud to architectures that run on traditional networks. If so, then cloud network architectures will display the same expected behavior as traditional network architectures, thus allowing the construction of networking testbeds at potentially substantial cost savings. Since the Amazon cloud does not support broadcast or multicast traffic, distributed applications face a challenge. Many distributed applications use broadcast or multicast to communicate real-time information. This research includes a case study for developing a distributed network application in the cloud which overcomes the restriction on broadcast and multicast traffic. During performance testing, the baseline network and cloud network configurations are provided statistically equivalent traffic workload. Metrics such as packet loss, delay, jitter and throughput are compared to determine relative performance. Analysis of the experimental results shows that in each case, the cloud network configurations performed at or above the performance level of the baseline network. Therefore, the public cloud infrastructure is suitable for performing integrated network experimentation. This research continues Project Everest's efforts to leverage cloud services for network experimentation. Project Everest is a framework which aims to combine emulation and cloud infrastructure into a single testbed using the Amazon Elastic Compute Cloud (EC2). Their tests indicate satisfactory cloud performance, but they recommend testing cloud network performance under various workload. This research carries out those performance tests.

Table of Contents

Page

Page

List of Figures

LEVERAGING THE CLOUD FOR INTEGRATED NETWORK EXPERIMENTATION

I. Problem Statement

1.1 Background

The Department of Defense (DoD) recognizes the need to create accurate representations of networking environments for the purposes of network planning, optimization, testing, and post-incident investigations. These *virtual networked environments* are distributed networks that allow communication between multiple systems in real time. Having a test network that is separate from the production network can be invaluable to network research and development however, it can also prove to be a costly venture. Therefore, researchers employ various methods to accomplish these objectives while minimizing required resources.

The traditional method to creating models of these environments uses discrete-event packet-level *simulation*. While this approach provides complete control, absolute repeatability of experiments, ease of use, execution efficiency, scalability, and lower relative cost, it comes at the cost of detail and realism [2]. For more realistic experiments, network emulation testbeds have been built out of real hardware networks consisting of hundreds of devices. *Emulation* combines real network elements and protocol implementations with simulated elements such as network links and background traffic. The primary difference is that simulation executes in virtual time, whereas emulation executes in real time. Although emulation is more realistic, it is not repeatable. *Integrated network experimentation* combines simulation and emulation into one experiment, each representing different portions of the topology [3].

Cloud computing platforms provide various resources to consumers at affordable prices such as storage, compute power, and email services [4]. Commercial cloud services provide reliable, scalable, and inexpensive computing platforms that can potentially meet the DoD's network modeling and experimentation needs. This research investigates the potential to use a cloud service such as Amazon Web Services (AWS) as a testbed for prototyping virtual networked environments using integrated network experimentation.

1.2 Statement of Problem/Issue

The goal of this research is to determine the feasibility of using the cloud to perform integrated network experimentation. That determination is based upon the performance of the cloud network relative to baseline network configurations under statistically identical workload conditions. Specifically, the goal is to make the following determinations:

- What techniques do researchers use to conduct network experimentation?
- What tools are used to implement these techniques?
- What effect on cloud testbed performance does physical distance between virtual machine (VM) instances have?
- What is the difference in packet loss, round-trip time, jitter and throughput among various Amazon cloud platforms, compared to a baseline platform?

1.3 Scope, Limitations, Assumptions

Amazon publishes specifications for VM instance types, but not the underlying hardware infrastructure that physically power these VMs. Consequently, users are unaware about details such as the number of VMs that are mapped to any single physical device, and the amount of available bandwidth between physical devices, among other details. The lack of knowledge regarding inter-region and intra-region link capacities, along with the latency and overhead associated with network management and virtualization poses a limitation.

Users can control the creation of VM instances, including their regional locations, but not the underlying hardware infrastructure itself. Therefore, although this research attempts to make determinations about the underlying infrastructure's influence on performance, this research focuses on the user-controlled aspects of the cloud configurations. Furthermore, this research does not consider security aspects of the cloud. Finally, this research assumes that Amazon's policies for network management and load balancing provide a viable platform to perform the desired testing.

1.4 Methodology

To determine if the cloud can adequately perform integrated network experimentation, network performance in the cloud is considered. This research measures the performance of four unique cloud platforms, and compares them to a baseline platform. Cloud performance is expected to not be statistically less than baseline performance. The factors that are expected to have the most impact on performance are the instance type, Amazon Availability Zone of virtual machine instances, packet size, packet arrival distribution and traffic protocol.

1.5 Overview of Thesis

The remainder of this thesis is organized as follows. The Literature Review discusses relevant theories, problems, and methodologies used in other published sources. Next, the Methodology describes the process of identifying and gathering field data. Chapter IV consists of data analysis and results. Additionally, this chapter interprets the validity, reliability and applicability of the results. Chapter V discusses a case study using distributed network applications in a cloud environment. Finally, the Conclusion summarizes the main points of the thesis, addresses the research questions and makes future research recommendations.

II. Literature Review

This chapter reviews literature from published sources that are most relevant to this research. The Introduction section provides a topic statement, explanation of key terms, justification for research and scope of the review. The Introduction section is followed by a review of network experimentation techniques used by NS-3, OPNET, Emulab, Planetlab, VINI, Live Network Testing, Everest and techniques implemented by this research. The Conclusion section summarizes the techniques and research contributions presented, and introduces the research methodology.

2.1 Introduction

This literature review explores various network experimentation techniques. Techniques are logically grouped under simulation, emulation, and other techniques. Information presented is a result of scholarly article searches, and reviewing each article's sources. The review is not meant to be exhaustive, but a survey of the most prominent techniques currently in use. Accurate models of networking environments facilitate network planning, optimization, testing, and post incident investigations. The cost of deploying a complete testbed containing multiple networked end hosts, intermediate devices and communications links can be prohibitive to many organizations. Consequently, network simulation and emulation are widely used experimental techniques that save time and money in accomplishing this task [5]. The remainder of this section explains simulation and emulation, providing baseline knowledge before proceeding with a review of the techniques.

Simulation uses discrete events to model real world networks at the packet level, providing repeatable network experimentation in a controlled environment [2]. Since every parameter of the experiment can be controlled, simulations can scale to large numbers of elements that can be programmed quickly. Additionally, simulation models are easy to configure and relatively inexpensive to modify, compared to changing a real network. These characteristics allow experimenters to build a rapid prototype-and-evaluate environment, capable of performing analysis of a wide variety of networking scenarios [2, 5]. Simulation provides this versatility to experimenters through the use of abstraction. Experimenters are able to choose the level of detail included in the simulation. As more detail is included within the simulation, the simulation more accurately reflects the real network. Conversely, as more details are abstracted away from the simulation, generally speaking, accuracy decreases, but the model is simplified and more efficient to execute. Thus, experimenters are forced to choose between efficiency and accuracy when deciding upon the simulation's level of abstraction without any systematic means of validating the choice of abstraction [6]. This presents challenges for experimenters. The abstraction level used in simulation may not allow experimenters to account for low level effects, or what might seem like small details, such as interrupts associated with heavy traffic loads. Furthermore, since simulators do not implement real protocols, they cannot integrate real networking elements into the simulation, thus sacrificing realism in the experiment.

Emulation combines real networking elements such as end hosts and protocol implementations with simulated elements such as background traffic, network links and intermediate nodes [2, 7, 8]. This technique allows experimenters to attach end systems to an emulator, and systems behave exactly as they would on a real network. By adding real-world interaction, emulation attempts to address the deficiencies of simulation, while retaining strengths such as repeatability and ease of configuration [7]. Emulation has generally taken on two forms: *network emulation* and *environment emulation*. Network emulation allows simulated network components to communicate with real-world protocol implementations. Environment emulation extends the concept further to build an operating system specific environment in which real-world protocol implementations are executed directly within the simulator [7]. A key difference between simulation and emulation is that simulation runs in virtual time, and emulation runs in real time [2]. Discrete-event computer simulators use events to advance time, meaning that the current time is immediately advanced to the dispatch time of the next pending event. Emulating real time is accomplished by modifying the base scheduler to not immediately advance to the time of the next pending event, but rather to dispatch that event at the appropriate real-time, by introducing a real-time delay [7]. In other words, events in real time are synchronized to the real world wall clock. By running tests on a testbed built out of real hardware, emulation allows experimenters to analyze applications that are affected by lower-level influences such as device interrupt handling, CPU scheduling or Network I/O [4]. Consequently, emulation makes integrated network experimentation possible. *Integrated network experimentation* combines real and simulated elements in the same experiment, each representing different portions of the topology, enabling new validation techniques and larger experiments than possible with real elements alone [2, 9]. The sections that follow describe simulation and emulation techniques.

2.2 Review of Network Experimentation Techniques

2.2.1 Simulation.

Various discrete-event network simulators exist in the networking research arena. Examples include OMNeT++, NetSim and GNS3, to name a few. This section highlights the most popular open source simulator, and the most popular proprietary simulator, ns-3 and OPNET respectively.

2.2.1.1 NS-3.

One of the most popular open source simulators, NS-3 [10], is widely used for research and education on both IP and non-IP based networks. The majority of its users employ NS-3 on wireless/IP simulations involving Wi-Fi, WiMAX or LTE, using protocols such as OLSR, OSPF, BGP and AODV for IP-based applications. The NS-3 simulator is written in C++ and Python [2], and features a modular, documented software core that allows user modification and customization. Additionally, it allows software integration, where users incorporate additional open source networking software, reducing the need to rewrite simulation models. Finally, NS-3 features virtualization support using lightweight virtual machines and protocol entities designed to be closer to real computers [5].

Because NS-3 is open source, everyone can contribute to it, customize it, find bugs and fix them. Though feature rich, NS-3 is not easy to use due to its lack of a graphical user interface (GUI). Users must type code to generate test scenarios, and understand concepts of queuing theory to correctly interpret the obtained results [2].

2.2.1.2 OPNET.

OPNET [11] is a commercial network simulation product presented by OPNET Technologies Inc. Its proprietary natures restricts its usage compared to NS-3. However, OPNET has many features not present in NS-3. It has a GUI to design and debug simulation scenarios, a fast discrete event simulation engine, various vendor device models to model specific networking scenarios, object-oriented modeling, and a fully parallel simulation kernel [2]. In addition to powerful visual or graphical support for users, parameters can be adjusted, and experiments can be repeated easily through the GUI [5].

2.2.1.3 Simulation Summary.

To summarize, here are the advantages and disadvantages of using simulators for networking experiments [4]:

<u>Advantages</u>

1. Simulation experiments are easy to set up and modify
2. Simulation gives the experimenter total control over the simulation and functionality of each element.
3. Simulation is cost effective, requiring less time through ease of use, and either zero cost (NS-3) or low-moderately expensive (OPNET).

Disadvantages

1. Like NS-3, many simulators lack a built-in GUI, requiring users to write code.

2. Simulation realism is limited by model detail, which is probably the most important disadvantage. Experiments requiring fine-grained measurements may give skewed results or cannot be produced. For example, modeling the traffic and congestion associated with Internet backbone links might not be possible if those effects are not included within a simulation model.

3. Some proprietary simulators may be expensive, thus the popularity of NS-3 in academia.

2.2.2 Emulation.

Just as there are numerous network simulators in use for network experimental research, there is no shortage of network emulators. This section highlights Emulab, Planetlab and VINI, the three most widely used emulation platforms.

2.2.2.1 Emulab.

Emulab is a platform that allows researchers to conduct integrated network experimentation on networks and distributed systems. It is a public facility available without charge to most researchers worldwide [9]. The environment integrates simulation, emulation and live network experimentation into a single framework, providing experimental control without sacrificing realism. These resources are available to researchers around the world free of cost [4]. Although there are over two dozen sites around the world, the primary Emulab installation is run by the Flux Research Group, part of the School of Computing at the University of Utah, consisting of hundreds of nodes of various configurations connected through rack switches [12]. Experimenters use a simple GUI to envoke an *ns* script to automatically configure a physical topology within the Emulab testbed, consisting of virtualized host names, IP Addresses, links and nodes [4]. Wide-area network (WAN) links are emulated within local-area network (LAN) environments by inserting a Dummynet node between two physical nodes to enforce queue and bandwidth limitations [9]. Emulab also contains simulation features that allow dynamic addition and removal of experimental nodes, and changing link characteristics at specific time instances during experiments [4].

Emulab has been widely used in research involving Active Networks, Adaptive Traffic Equalization, Cloud Computing, Databases, and many more, but lack of resources is a primary concern since more experiments exist than available hardware [13, 14]. Resource shortage along with database software and hardware errors are some of Emulab's challenges [14].

2.2.2.2 PlanetLab.

PlanetLab is a networking and distributed systems research testbed, consisting of thousands of geographically distributed nodes worldwide [15, 16]. Most machines are hosted by research institutions, and all are connected to the internet, giving PlanetLab the realism of having traffic flow through the actual Internet between nodes that peer with the Internet's regional and long haul backbones [16]. Accounts are limited to persons affiliated with the universities and corporations that host PlanetLab nodes, along with a limited number of free public services such as a proxy server system and a peer-to-peer content distribution network.

PlanetLab is an overlay testbed, meaning that every application has a *slice* of its resources. This *Internet in a Slice (IIaS)* construct allows node allocation in various geographical locations [4]. Because all nodes are connected to the Internet, applications experience congestion, failures and diverse link behaviors associated with traffic flowing over the Internet backbone [16].

2.2.2.3 VINI.

VINI is a virtual network infrastructure that supports experiments involving simultaneous arbitrary network topologies on a shared physical infrastructure [17]. VINI is implemented on the Planetlab infrastructure, and it enables experiments that run real routing software, creating real network conditions, controlling network events, and carrying real traffic. Each node in VINI is a virtual machine (VM) that provides a realistic abstraction of a real machine, being able to run various operating systems and modifying its kernel, device drivers, and other subsystems [4]. VINI uses XORP routing [18] and Click forwarding [19] along with OpenVPN to build virtual topologies that allow arbitrary end hosts to direct traffic through the VINI infrastructure. VINI also uses network address translation (NAT) to ensure that traffic returns through VINI. Overall, VINI provides experimental traffic control by setting up routing tables on virtual routers, and directing traffic through the actual Internet for realism.

2.2.2.4 Emulation Summary.

To summarize, here are the advantages and disadvantages of using emulators for networking experiments [4]:

Advantages

1. Emulation provides greater realism and control through lower levels of abstraction and real hardware. Experimenters are able to specify arbitrary topologies and control traffic flow.

2. Emulation executes real software on nodes, as they are virtual machines.

3. Emulation uses simulation techniques to mimic components, such as simulating a 1Gbps link to appear as a 100Mbps link.

Disadvantages

1. Despite virtualization techniques, experimenters still face the possibility of the number of experiments exceeding available resources.

2. Managing and maintaining emulation testbeds is an expensive endeavor. Human operators may be required when remote troubleshooting is not possible.

3. Because emulation provides realism, this reduces experimental controls and makes it almost impossible it is almost always impossible to reproduce an exact scenario.

2.2.3 Other Methods.

2.2.3.1 Live Network Testing.

Live Network Testing is using existing Internet Links to run experiments. This method is not as popular because of the inability to reproduce scenarios, unpredictable link qualities and lack of control over intermediate routers and switches [4].

2.2.3.2 Everest.

Everest is a framework which aims to combine emulation and cloud infrastructure into one testbed using Amazon EC2, providing realism and experimental control [4]. The Everest infrastructure consists of a private cluster with a limited number of machines operated by The ONE research group at Carnegie Mellon University. Their preliminary research consists of performance tests comparisons between the private cluster and an identical topology hosted in the Amazon EC2 cloud [4]. Everest's advantages include realism and control of network topology, VMs, routing protocols, physical infrastructure in the private cluster, and the ability to integrate real hosts connected to the internet into experiments. Disadvantages include lack of a GUI to configure experiment topologies, difficulty in debugging routing parameters, and no access to Amazon's underlying infrastructure.

2.3 Experimental Resources

This section describes the resources employed while conducting the research. It begins with an introduction to Amazon Web Services (AWS), which is the public cloud infrastructure that is used to construct the testbed and conduct performance testing. AWS is followed by a description of the Distributed Internet Traffic Generator (D-ITG), which generates the traffic loads for the performance tests, and records experimental metrics.

2.3.1 Amazon Web Services (AWS).

Amazon offers a variety of services under the umbrella of Amazon Web Services (AWS). This research uses the Amazon Elastic Compute Cloud (Amazon EC2). Amazon EC2 [20] provides resizable compute capacity in the cloud, making web-scale computing easier for developers. This platform allows users to rent computing power to easily scale up and down to meet business resource needs. Amazon caters to small and medium businesses, allowing them to rapidly deploy applications and services ranging from distributed applications to social networking applications to large scale cloud storage systems [4]. Amazon makes computing resources available through *EC2 instances*, which are virtual machines that allow users to specify instance types with various amounts of virtual CPUs, memory, storage space, and processor. Additionally, Amazon defines a metric called EC2 Compute Units (ECU) to provide a more standardized comparison among instance types. Finally, users can choose the Amazon regional location of the instance, Availability Zone within that region and the operating system to install on the instance [20]. Some of the main advantages of Amazon EC2 include [4]:

1. Rapid Deployment: An EC2 instance can be configured in just a few minutes, allowing users to rapidly create many VMs.

2. Superuser Privileges on Instances: Users have root access within the EC2 instances to install software, modify kernel settings, or any of the low-level subcomponents.

3. Scalable Solutions: Businesses can build scalable solutions for their clients, ranging from webhosting to billing services to social networking location check-ins.

Although the Amazon cloud has many benefits, it also has challenges. Users are allowed to create instances in the cloud, but are not allowed to control the underlying networking infrastructure that connects instances. This limits the administrative control over setting up the network topology in the cloud testbed. Additionally, users cannot control the amount of bandwidth between nodes. Amazon does not publish performance data regarding its underlying infrastructure, so users must trust that sufficient bandwidth exists to support their networking scenario. The same is true regarding simulating or emulating poor network links. Users may have the need to create a topology that includes smaller and less reliable network links in order to test application performance in adverse conditions. The cloud does not have a native way to simulate or emulate link capacities. Here are some of the notable issues with Amazon EC2 [4]:

1. SLA Guarantees: SLA guarantees a monthly uptime percentage of at least 99.95 percent for Amazon EC2 and Amazon EBS within a region, but does not mention any network performance guarantees.

2. No Access to Physical Infrastructure: Users are not allowed access to the physical infrastructure, and Amazon has not made information about the physical infrastructure public. Therefore, users must rely on Amazon's virtualization and load balancing techniques to manage the load on their data centers.

3. Outages: Since the Amazon cloud does not fall under the user's administrative control, users must rely on Amazon to resolve any outages to the cloud infrastructure.

4. Broadcast and Multicast Traffic: Amazon does not allow broadcast or multicast traffic in the cloud. This may affect applications that rely on this form of communication.

2.3.2 Distributed Internet Traffic Generator (D-ITG).

In order to determine the suitability of the Amazon cloud as a DoD networking testbed, cloud performance under various workloads must be considered. Networks carry diverse traffic patterns with a mixture of protocols, so the workload generator must reflect this diversity. The Distributed Internet Traffic Generator (D-ITG) is a platform capable to produce packet level traffic that accurately replicates appropriate stochastic processes for both Inter Departure Time (IDT) and Packet Size (PS) random values. These values can assume several distributions that include normal, Pareto, uniform, exponential and Poisson, among others [21]. Additionally, D-ITG supports IPv4 and IPv6 traffic at the network, transport and application layer. Finally, D-ITG supports Linux, Windows, OSX and FreeBSD operating systems [21]. These capabilities plus its free and open source nature makes D-ITG an ideal choice as a workload generator.

2.4 Research Contributions

This research contributes to the body of knowledge in the following two ways:

1. This research performs extensive cloud performance testing under various workload. Experimental factors include Virtual Machine (VM) instance type, Amazon Availability Zones, traffic protocol, packet size and packet distribution. This research uses performance metrics to determine if architectures constructed in the cloud perform as well as architectures that run on traditional networks. If so, then cloud network architectures will display the same expected behavior as traditional network architectures, allowing the construction of networking testbeds at potentially substantial cost savings.

2. Since the Amazon cloud does not support broadcast or multicast traffic, distributed applications face a challenge. Many distributed applications use broadcast or multicast to communicate real-time information. In many cases, moving a distributed application into a cloud computing environment demands the use of different protocols. In the case study for developing a distributed network application in the cloud, this research identifies a messaging framework for distributed applications in the cloud, which overcomes the restriction on broadcast and multicast traffic.

2.5 Conclusion

2.5.1 Summary of Simulation and Emulation Techniques.

Simulation and emulation are widely used network experimentation techniques. Discrete-event simulators such as ns-3 and OPNET make experiments easy to set up, modify, and control, but cannot provide the realism of emulation. Emulation platforms such as Emulab, PlanetLab and VINI provide realism and greater experimental control, but have limited resources, and experiments are not deterministic. The Everest project attempts to address emulation shortfalls by leveraging the cloud. However, Everest has neither integrated the cloud into its private infrastructure to address the resource shortfall, nor conducted extensive cloud performance testing under different loads to assess the feasibility of a testbed hosted entirely in the cloud.

2.5.2 Research Contributions Summary.

This research continues Everest's efforts to create a cloud networking testbed by conducting extensive performance testing under different loads. Additionally, it suggests a framework for distributed applications to overcome the cloud's policy of restricting broadcast and multicast traffic. Together, it determines if and how integrated network experimentation can be conducted entirely in the cloud to support various networking and distributed application scenarios.

III. Methodology

This chapter provides a methodology to evaluate cloud testbed performance. The problem definition is presented, including the goals, objectives, and the approach to achieving those goals. The sections that follow discuss the system boundaries, system services, workloads, performance metrics, system parameters and factors, followed by the evaluation technique and experimental design. The Data Interpretation section describes the statistical method for comparing mean values between the baseline platform and each of the four cloud platforms. Finally, the summary reviews the main points presented in the chapter.

3.1 Problem Definition

3.1.1 Goals.

The goal of this work is to determine the feasibility of using the cloud to perform integrated network experimentation. Specifically, the goal is to:

1. Determine the effect on performance of physical distance between VM instances.
2. Determine the difference in packet loss, round-trip time, jitter and throughput among various Amazon cloud platforms that are emulating a local area network (LAN). Compare this to a virtualized baseline platform located on a single host.

3.1.2 Hypothesis.

Figure 3.1 shows an illustration of this research's hypothesis. The null hypothesis is that the mean values for packet loss, delay, jitter and throughput for each of the four cloud platforms are not statistically different than that of the baseline platform. The alternate hypothesis is that the mean values for packet loss, delay, jitter and throughput are statistically different than that of the baseline platform.

$$H_o: \mu_{baseline} = \mu_{cloud1,2,3,4}$$
$$H_a: \mu_{baseline} \neq \mu_{cloud1,2,3,4}$$

Figure 3.1: Hypothesis

3.1.3 Assumptions and Limitations.

Amazon does not publish performance metrics for the underlying hardware that supports its cloud computing platform. This absence of information regarding inter-region and intra-region link capacities, latency, and overhead associated with network management and virtualization is a limitation of this research. Users only control the creation of virtual machine (VM) instances, including the regional location of the VM, but not the underlying hardware infrastructure itself. This research assumes that Amazon's policies for network management and load balancing provide a viable platform for the desired testing.

3.1.4 Approach.

This research takes the following approach to achieving the stated goals and testing the hypothesis. The baseline platform and the various cloud platforms are provided statistically identical workloads. The baseline platform serves as a standard to which to compare the performance of the cloud platforms. Differences in performance metrics are identified and analyzed. The suitability of various platforms for integrated networking experimentation is assessed. If differences are identified, suitable data transformation processes are developed, if possible, to translate or normalize data from the emulated environment to be statistically not different than baseline performance.

3.2 System Boundaries

The system under test (SUT) is the network emulation environment. The component under test (CUT) is the Amazon cloud. Figure 3.2 depicts the SUT and CUT.

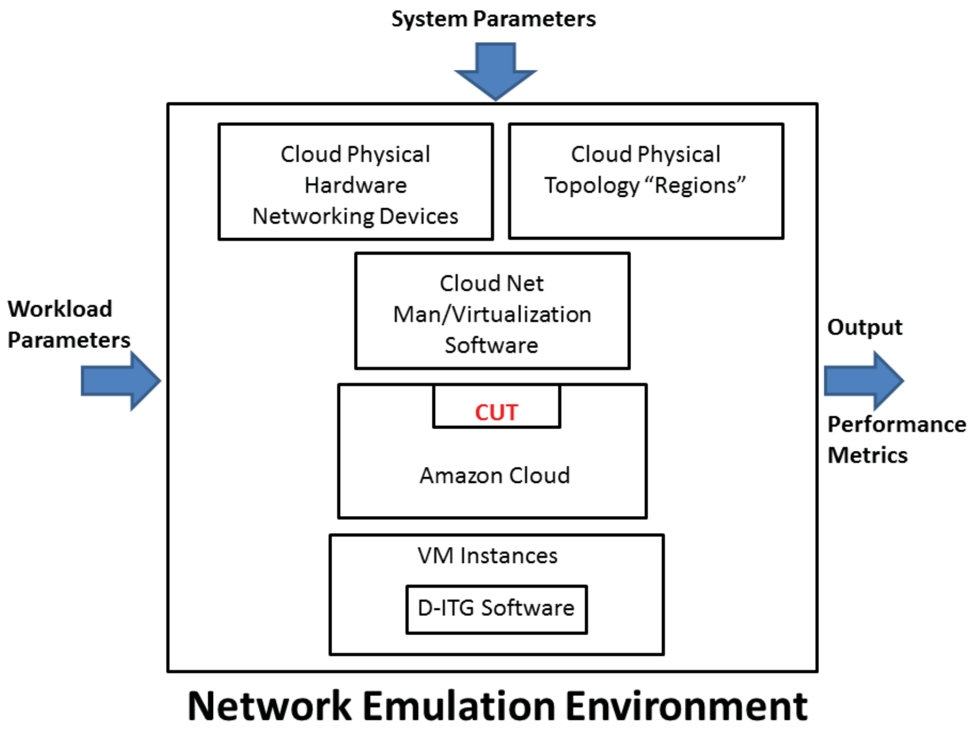

System Parameters
Workload Parameters
Cloud Physical Hardware Networking Devices
Cloud Physical Topology "Regions"
Cloud Net Man/Virtualization Software
CUT
Amazon Cloud
VM Instances
D-ITG Software
Output
Performance Metrics
Network Emulation Environment

Figure 3.2: System Under Test (SUT) and Component Under Test (CUT)

3.3 System Services

The system provides a network emulation service. Success is defined as the extent to which the performance of the network emulation environment meets or exceeds the performance of the baseline platform. If the emulation service is successful, its performance will not be statistically less than that of the baseline platform. If the performance is statistically worse than that of the baseline platform, then failure has occurred.

3.4 Workload Parameters

The workload for the system is the traffic presented to the SUT for transport. Network traffic comes in many forms. Some examples include Internet traffic, distributed applications, and virtual private networks. These traffic workloads are diverse in size, quantity, and distribution. Additionally, the workload may change over time based upon social behavior, technology, and many other considerations. It is imperative that experimental workloads mimic these characteristics in order to create more realistic scenarios for the SUT. The Distributed Internet Traffic Generator (D-ITG) generates workloads for the SUT. D-ITG is capable of generating traffic based upon protocol, packet size, payload and many other attributes. Additionally, the program is capable of measuring many performance indicators, including one-way delay, round trip time, jitter, packet loss and throughput [22]. The workload parameters include:

1. Packet Size - Packet size has an affect on network performance. For example, at a constant rate of 1,000 packets per second, the throughput of a network link that is transferring 512 Byte packets is expected to be higher than a network link transporting 256 Byte packets, all else being equal.

2. Packet Arrival Distribution - The packet arrival distribution refers to the manner in which transmission requests are sent to the system. Offered load arriving at a constant rate as opposed to Poisson arrivals may have performance implications.

3. Traffic Protocol - Traffic containing the following protocols will be sent to the System Under Test:

   - TCP - Transmission Control Protocol (TCP) is a reliable transport layer protocol that operates on top of the best effort Internet Protocol (IP) layer protocol to facilitate host to host communication. To perform services such as reliable transmission, error detection, flow control and congestion control, TCP requires additional overhead. Consequently, there is a trade off between the benefits that these services provide and their associated performance costs.

- UDP - In contrast to TCP, the User Datagram Protocol (UDP) is an unreliable transport layer protocol that operates on top of IP. It is meant to be a lightweight protocol that does not include the overhead associated with TCP. Because UDP has less overhead, its throughput performance should be better than TCP.
- ICMP - This experiment uses the Internet Control Message Protocol (ICMP) to measure round-trip time (RTT) between two nodes.

4. Workload Generator Random Seed - The D-ITG software has nine unique seeds available. These seeds are altered randomly to ensure that workloads presented to the SUT are not identical among test runs.

3.5 System Parameters

The parameters discussed below affect the performance of the cloud testbed.

1. Operating System - All virtual machines run Linux Ubuntu.
2. Processor - All virtual machines run a 64 bit processor.
3. Number of Virtual CPUs - The number of virtual CPUs that a VM has varies by instance type. Baseline VMs will have one virtual CPU.
4. Number of Amazon EC2 Compute Units (ECU) - ECUs are defined by Amazon to make it easy to compare CPU capacity between different instance types. This parameter does not apply to the baseline network.
5. Memory - The amount of memory available varies with VM instance type. Baseline VMs have 1 GB of memory.

6. Storage - VM storage capacity varies with VM instance type. Baseline VMs have 410 GB of storage.

7. Network Resource Reservation - Amazon makes network performance promises based upon VM instance type that includes very low, low, moderate, high and 10 Gigabit. These promises are not included in the SLA, but they serve as a means of comparing instance types. This parameter does not apply to the baseline network.

8. Amazon Region - The Amazon cloud consists of nine different regions across the world, which are isolated from other regions. This research examines whether network performance within one region differs from other regions. This parameter does not apply to the baseline network.

9. Amazon Availability Zone - The assumption behind this parameter is that the physical distance between hosts affects network performance. Within each region, Amazon separates its networks into availability zones. Amazon does not publish specific details regarding the underlying network infrastructure, but they do say that virtual machine instances in different availability zones are in physically separate locations. This parameter does not apply to the baseline network.

10. Time of Day - The key assumption regarding this parameter is that cloud datacenter utilization is not evenly distributed throughout the day. This research attempts to represent that by conducting experiments during certain time frames that correspond to various utilization levels. This parameter does not apply to the baseline network.

11. Performance of Underlying Amazon Hardware - Amazon's underlying hardware provides the computational power in the cloud environment. Amazon does not publish specific numbers regarding this infrastructure. However, this is a parameter because its performance has an effect on the overall performance of the cloud environment. For example, the range of available bandwidth between network

nodes within the same availability zone, and between different availability zones within a region may limit performance for a large distributed topology in the cloud. This research assumes that Amazon's underlying hardware can sufficiently support integrated network experimentation.

12. Performance of Amazon Network Management/Virtualization Sofware - In addition to the underlying hardware, Amazon's network management and virtualization software also plays a role. For example, the amount of virtual machines mapped to a single physical node may affect network performance for cloud platforms relying on this software. This research assumes that Amazon's load balancing and virtualization scheme can sufficiently support integrated network experimentation.

3.6 Performance Metrics

The following performance metrics are used to evaluate the performance of the SUT for a given platform.

- Round-Trip Time - Round-trip time (RTT) is the length of time in milliseconds it takes for a packet to be sent from source to destination, plus the time to receive an acknowledgement. This metric captures transmission, propagation and queuing delays present on the network between the source and destination nodes. RTT considers only packets that are successfully delivered. The measurement is usually determined using the Packet Internet Groper, or ping utility, which uses ICMP. This metric provides insight into the quality of successful packet delivery.

- Packet Loss - Packet loss is the percentage of packets that do not successfully arrive at their destination out of the total number of packets sent. This metric measures the failure outcomes during transmission.

- Delay - This metric captures the end-to-end one-way delay for packets between source and destination nodes. This metric is measured in milliseconds, and only considers packets that are successfully delivered, providing insight into the stability of packet delivery performance.

- Jitter - In this experiment, jitter is defined as the packet delay variation (PDV). RFC 3393 defines PDV as the difference in end-to-end one-way delay between selected packets in a flow, ignoring lost packets. This metric only considers packets that are successfully delivered, and provides further insight into the stability of packet delivery performance.

- Throughput - One of the most common ways of measuring network performance is by measuring throughput. Throughput is the rate at which messages are successfully delivered over a communication channel. Therefore, throughput only considers entire packets that successfully arrive at their destination. In this experiment, throughput is measured in Megabits per second.

3.7 Factors

Appendix A describes the process used to choose experimental factors from the list of experimental parameters. Table 3.1 shows the experimental factors used in this research and their corresponding levels.

3.8 Evaluation Technique

This research uses direct measurement to evaluate the performance each Amazon cloud platform. Four distinct cloud platforms are constructed along with a baseline platform. D-ITG software is used to generate experiment traffic, and gather performance metrics during the experiments. Sender and receiver VMs run D-ITG software. Results from experimental runs in the cloud are validated against the results from the baseline platform. The platforms are set up as follows:

Table 3.1: Experimental Factors with Corresponding Levels

Factor	**Low Level**	**Mid Level**	**High Level**
1. Instance Type	M1.Medium		M1.Large
2. Amazon Availability Zone	Same AV Zone		Different AV Zone
3. Traffic Protocol	ICMP	UDP	TCP
4. Packet Size Distribution	Constant Size	Poisson	Uniform
(Measured in Bytes)	512	Mean=512	Min=256, Max=4096
5. Packet Arrival Distribution	Uniform	Poisson	Constant Rate
(Measured in Packets/sec)	Min=256, Max=4096	Mean=1,000	1,000

- LAN Baseline - The baseline platform consists of two VMs on the same local host. VMs are hosted on the same physical machine using virtualization software such as VMWare. Each VM runs the D-ITG software, and sends traffic to the other VM as shown in Figure 3.3.

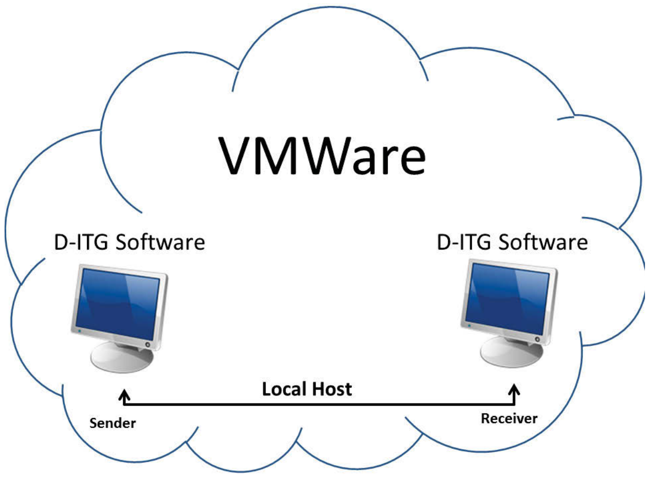


Figure 3.3: Baseline Platform

- Two VM Instances in the same Availability Zone - Figure 3.4 shows a cloud platform with two VM instances within the same availability zone. Two of the four cloud platforms use this construct, one with M1.Medium instances and one with M1.Large instances. These platforms are compared to the LAN baseline platform.

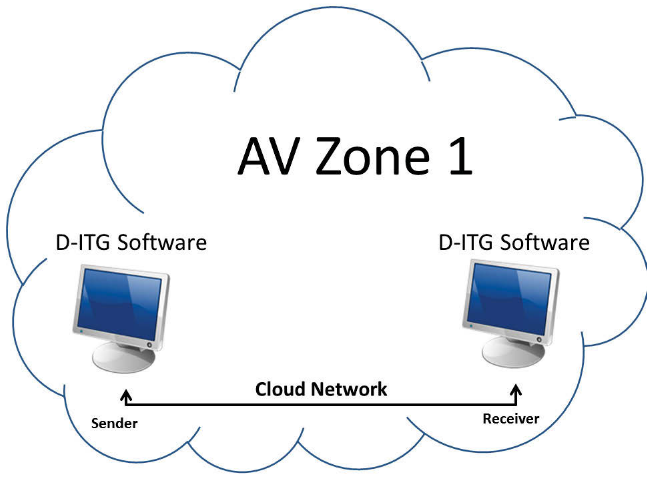


Figure 3.4: Cloud VMs in Same Availability Zone

- VM Instances in Different Availability Zones - Figure 3.5 shows a cloud platform with two VM instances located in different availability zones within the same region. Two of the four cloud platforms use this construct, one with M1.Medium instances and one with M1.Large instances. These platforms are also compared to the baseline platform to capture the performance effects of geographical separation.

3.9 Experimental Design

To determine the factors, a Plackett-Burman [23] design is used to evaluate the effects of each parameter. There are four workload parameters, and 12 system parameters, for a total of 16 experimental parameters. However, only seven of them can be independently varied, so there are seven potential experimental factors. Under these conditions, the

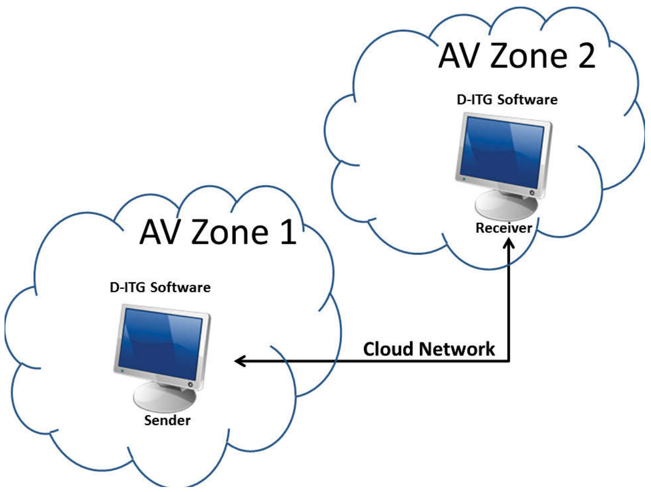
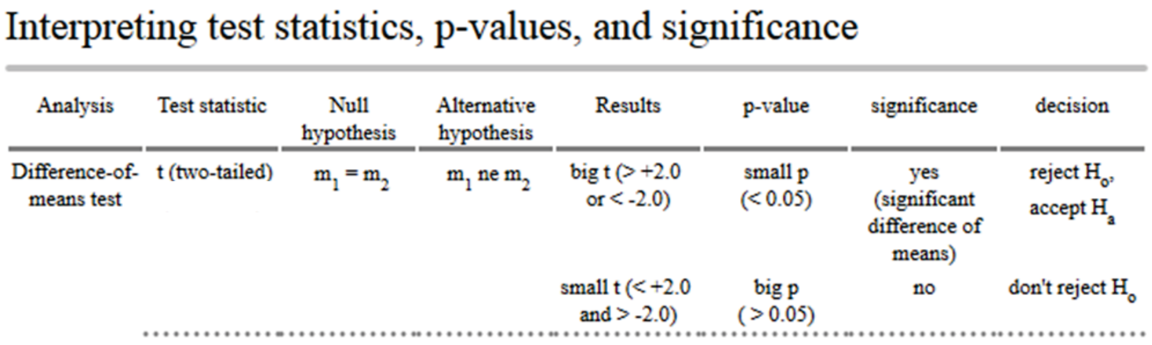


Figure 3.5: Cloud VMs in Different Availability Zones

Plackett-Burman design requires 12 experimental runs for the screening process. The ICMP tests are conducted using the PING utility, requiring a total of 30 experiments. For the remaining experiments, the five most influential factors are considered. To evaluate the interaction among all the factors, a full factorial design is used. There are five factors that each have various levels. A full factorial design requires 2x2x3x3x3= 108 experiments. Sufficiently small variance is expected with no more than five replications, resulting in a total of 540 experiments. Therefore, the overall number of experiments including Plackett-Burman, PING and all remaining experiments is 12+30+540= 582 experiments. Each test runs for a 10 second time period. The random seed within the workload generation software is changed before each run to ensure each is independent. Errors are assumed to be normally distributed, and a 95 percent confidence interval is used because cloud performance is expected to mimic baseline performance.

3.10 Data Interpretation

As part of the data analysis process, t-tests are run on the data to determine if a difference of means exists. The null hypothesis for each experiment in this chapter is that the means are equal, and the alternative hypothesis is that they are not equal. Figure 3.6 shows a breakdown of the process to determine which hypothesis to accept. In instances with values for the t-statistic that have an absolute value greater than 2.0 along with p-values less than 0.05, there is a significant difference of means. Therefore, the null hypothesis is rejected, and the alternative hypothesis is accepted. In instances with values for the t-statistic that have an absolute value smaller than 2.0 along with p-values larger than 0.05, there is no significant difference of means. Therefore, the null hypothesis is not rejected.



Interpreting test statistics, p-values, and significance

Analysis	Test statistic	Null hypothesis	Alternative hypothesis	Results	p-value	significance	decision
Difference-of-means test	t (two-tailed)	$m_1 = m_2$	m_1 ne m_2	big t (> +2.0 or < -2.0)	small p (< 0.05)	yes (significant difference of means)	reject H_o, accept H_a
				small t (< +2.0 and > -2.0)	big p (> 0.05)	no	don't reject H_o

Figure 3.6: Statistical Data Analysis (reprinted from [1])

3.11 Methodology Summary

To determine if the cloud can suffice testbed for prototyping virtual networked environment architectures, network performance in the cloud is considered. This research measures the performance of four cloud platforms, and compares them to a baseline platform. Cloud performance is expected to not be statistically less than baseline performance. The factors that are expected to have the most impact on performance are the Amazon Availability Zone of VMs, VM instance type, packet size, packet arrival distribution and traffic protocol.

IV. Data Analysis and Results

This chapter discusses the data from experimental runs, analyzing data gathered using the PING utility, and workload traffic generated using the Distributed-Internet Traffic Generator (D-ITG). Data presented in this chapter consists of calculated mean values of six experimental replications. The D-ITG traffic workload data shows the calculated mean values after six replications of each of the 18 unique traffic workload configurations. Individual measurements from each experiment are found in the appendixes. Appendix B shows the raw measurements across the baseline platform, and Appendix C shows the raw measurements across each of the four cloud platforms. When evaluating network performance, several metrics are considered:

1. Bandwidth - Bandwidth is the maximum amount of raw data that can be transmitted per second across a network link. Due to the nature of virtualization and the unknown details regarding Amazon's underlying infrastructure, this study is unable to calculate the amount of available bandwidth between two virtual machines. Therefore, bandwidth is not a metric used.

2. Latency - Latency is the minimum time a network needs to send the smallest possible amount of data. This value captures all processing, queuing, transmission and propagation delays present on the network. Unlike network delay, this does not take traffic workload into account. Therefore, latency describes travel time strictly associated with traversing the network under ideal conditions. This study uses the PING utility to capture the round-trip time (RTT) as a metric.

3. Packet Loss - Packet Loss is the percentage of packets that are not successfully delivered from source to destination, out of the total number of packets transmitted.

This includes packets that are dropped, lost, or corrupted along the way. This study uses Packet Loss as a metric for experiments involving traffic workload.

4. Delay - While the PING utility captures the latency involved with sending the smallest possible amount of data, it does not describe network performance under heavy traffic workload. Network Delay is a metric used by the study to capture the processing, queuing, transmission and propagation delays during traffic workload.

5. Jitter - Jitter is the packet delay variation (PDV). RFC 3393 defines PDV as the difference in end-to-end one-way delay between selected packets in a flow, ignoring lost packets. This study uses Jitter as a metric to capture PDV during traffic workload.

6. Throughput - Throughput is the actual data that is transmitted per second, excluding the necessary overhead used to send that data. throughput depends on many other factors such as the amount of bandwidth, latency, payload size, packet size, number of intermediate devices between source and destination, an others. This study uses Throughput as a metric for experiments involving traffic workload.

The rest of the chapter is organized as follows. The discussion begins with an analysis of the data captured by the PING utility. The PING experiments are used to get a general idea of underlying network performance, such as identifying whether any inherent problems appear on the surface of the network before introducing network traffic with realistic payloads. Next, traffic workload data is analyzed. The traffic workload experiments go into further detail by introducing various patterns of traffic, representing the bulk of performance analysis. The chapter concludes with a summary of findings, revisiting of investigative questions, and recommendations for future research.

4.1 PING Data Analysis

This section presents the results of running the PING utility between two hosts on the baseline network, as well as two hosts within each of the four cloud computing platforms. PING times correlate very roughly with the amount of distance between source and destination machines. A machine can PING itself very quickly, but it takes longer to PING other machines on the network. This time increases as distance increases. For example, a PING cannot exceed the speed of light. The distance between New York and Los Angeles is roughly 2,462 miles, which can be traversed by light in roughly 13 milliseconds one-way. A PING utility that travels this distance should not report a round-trip time (RTT) shorter than 26 milliseconds. Here are a few examples of latency sensitive systems that have performance thresholds:

1. Satellite Telephony - Geosynchronous telecom satellites are at least 71,000 kilometers from transmitter to receiver, and the resulting latency is roughly 473 milliseconds [24]. Regardless of available bandwidth, this can be very noticeable, and affect the quality of the satellite phone service.

2. World Wide Web - When delays are less than 100 milliseconds, Internet users feel that responses are instant from click to response [25].

3. Online Gaming - Real-time, multi-player games may use the internet or a local area network (LAN) to create a shared environment between two or more users. The maximum latency tolerance varies from game to game, but in general, first-person shooter games require lower latency for the best experience, while a turn-based game such as spades can tolerate higher latency [26].

The systems listed above are a subset of experiences that are affected by network latency. In order to accommodate a diverse range of systems that might appear on a network, latency should be as low as possible. Figure 4.1 shows the average latency values captured by the PING utility for the baseline platform and all four cloud platforms. PING statistics are summarized below:

1. Baseline Platform - The baseline platform has an average latency of 0.26 milliseconds, and 0 percent packet loss.

2. Cloud Platform 1 - Cloud Platform 1 has an average latency of 0.62 milliseconds, and 0 percent packet loss.

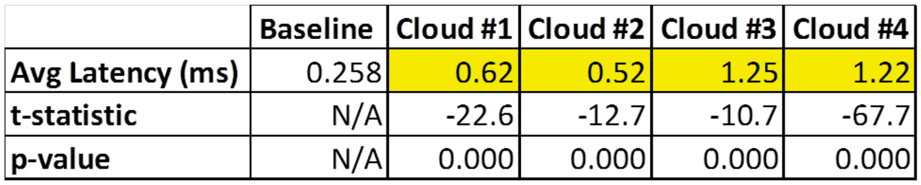
3. Cloud Platform 2 - Cloud Platform 2 has an average latency of 0.52 milliseconds, and 0 percent packet loss.

4. Cloud Platform 3 - Cloud Platform 3 has an average latency of 1.25 milliseconds, and 0 percent packet loss.

5. Cloud Platform 4 - Cloud Platform 4 has an average latency of 1.22 milliseconds, and 0 percent packet loss.

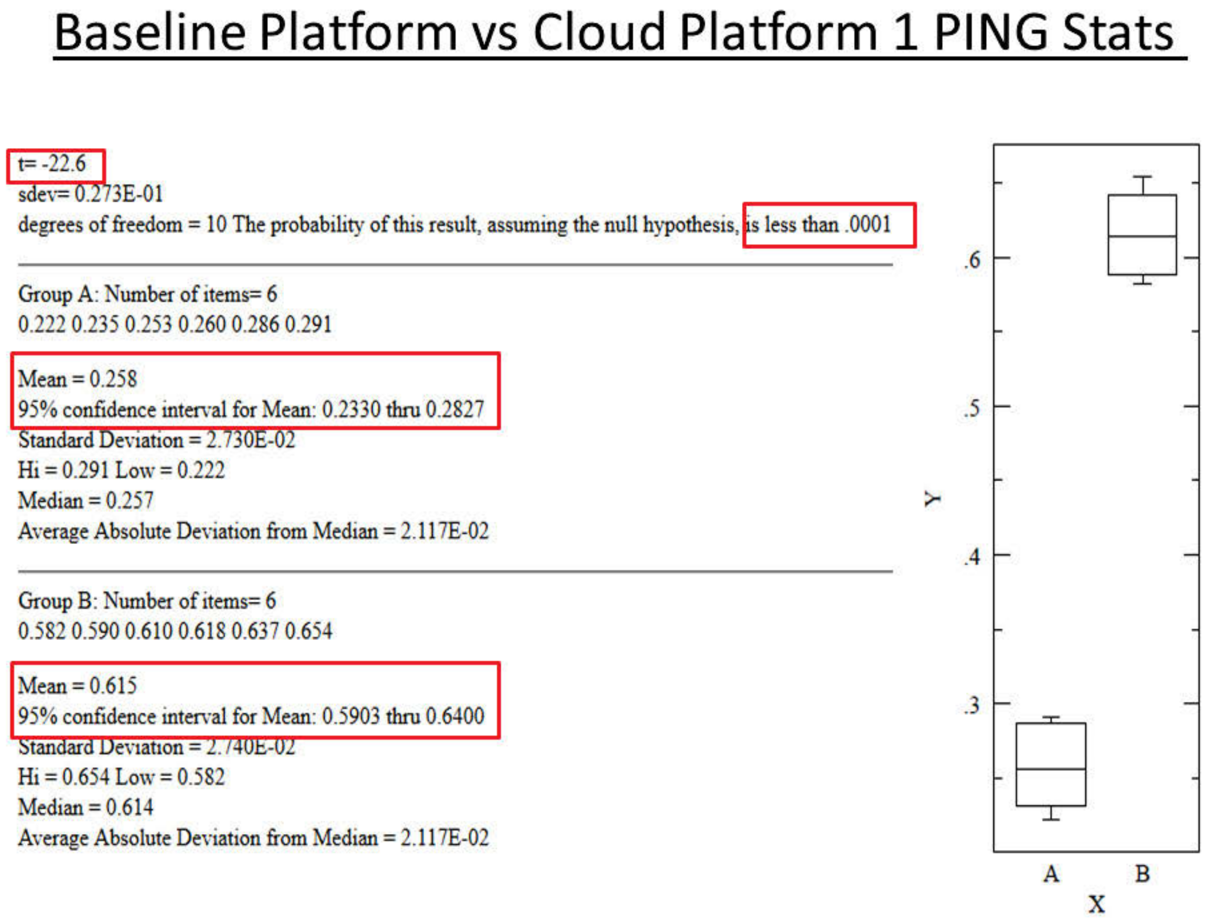
Figure 4.2 shows an example of interpreting the t-test results, comparing PING data between the baseline platform and Cloud Platform 1. The t-statistic of -22.6 indicates a higher absolute value than 2.0, and the p-value is less than 0.001, which indicates that there is a statistical difference. Therefore, the null hypothesis is rejected in favor of the alternative hypothesis. The 95 percent confidence interval for the mean on the baseline platform is 0.2330 through 0.2827. The 95 percent confidence interval for Cloud Platform 1 is 0.5903 through 0.6400. Notice that these confidence intervals do not overlap. The box plot on the right shows a visual depiction of the data, clearly showing no overlap of the confidence intervals. This is always the case when we see t-statistics with a larger absolute value than 2.0, and a p-value less than 0.05. In cases where the absolute value of the t-statistic is less than 2.0 and the p-value is greater than 0.05, the 95 percent confidence intervals overlap, and the null hypothesis cannot be rejected.

The high t-statistic values and low p-values in all cases indicate that the difference in the average latency values are statistically significant. Therefore, the null hypothesis



	Baseline	Cloud #1	Cloud #2	Cloud #3	Cloud #4
Avg Latency (ms)	0.258	0.62	0.52	1.25	1.22
t-statistic	N/A	-22.6	-12.7	-10.7	-67.7
p-value	N/A	0.000	0.000	0.000	0.000

Figure 4.1: PING Results



Baseline Platform vs Cloud Platform 1 PING Stats

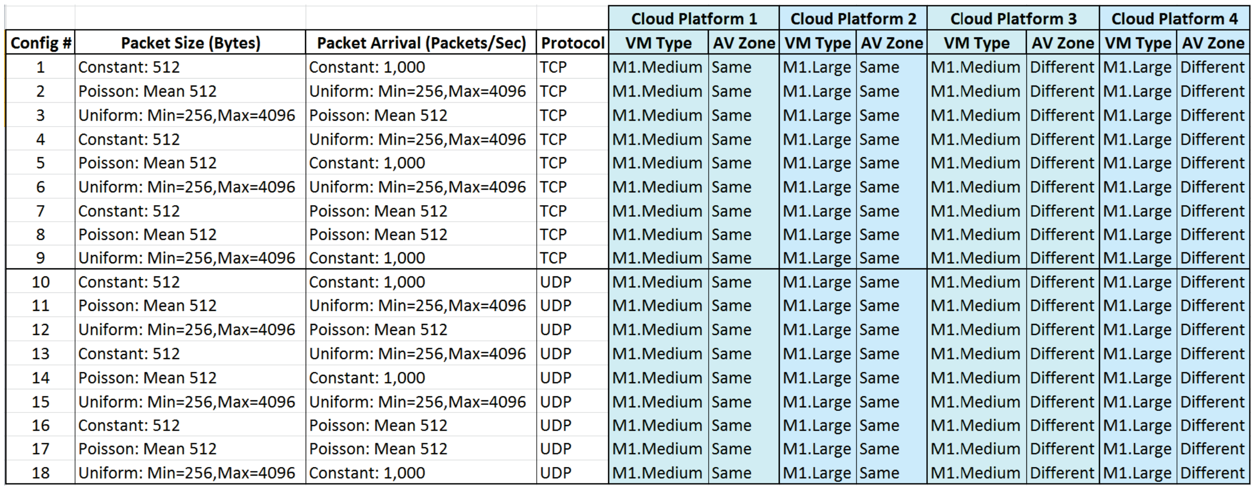
```
t= -22.6
sdev= 0.273E-01
degrees of freedom = 10 The probability of this result, assuming the null hypothesis, is less than .0001

Group A: Number of items= 6
0.222 0.235 0.253 0.260 0.286 0.291

Mean = 0.258
95% confidence interval for Mean: 0.2330 thru 0.2827
Standard Deviation = 2.730E-02
Hi = 0.291 Low = 0.222
Median = 0.257
Average Absolute Deviation from Median = 2.117E-02

Group B: Number of items= 6
0.582 0.590 0.610 0.618 0.637 0.654

Mean = 0.615
95% confidence interval for Mean: 0.5903 thru 0.6400
Standard Deviation = 2.740E-02
Hi = 0.654 Low = 0.582
Median = 0.614
Average Absolute Deviation from Median = 2.117E-02
```

Figure 4.2: PING Results: Baseline vs Cloud 1

is rejected and the alternative hypothesis is accepted for latency performance. Since the baseline platform consists of two virtual machines on the same host, its lower level of latency compared to all cloud platforms makes sense. It also makes sense that Cloud Platforms 1 and 2, which have virtual machines located within the same availability zone, have lower latency numbers than Cloud Platforms 3 and 4, which have virtual machines located in different availability zones. These numbers illustrate the affect that physical distance has on network latency. Although each cloud platform displays higher network latency than the baseline network, all latency values are well within any reasonable threshold. Therefore, there are no symptoms on the surface of the cloud network to suggest performance issues.

4.2 Traffic Workload Data Analysis

Workload tests provide greater insight to network performance by introducing significant traffic payloads across the network. This section conducts a side-by-side comparison of performance data between the baseline network and each of the cloud platforms. For each experimental run, each platform experienced statistically equivalent traffic workload, allowing for fair performance comparison. Figure 4.3 shows the 18 traffic workload configurations. Each configuration has a standard traffic pattern to provide a means for comparison across platforms. For example, Configuration 1 consists of traffic that has 512 byte packets, flowing at a constant rate of 1,000 packets per second using the TCP protocol. This is standard across all four cloud platforms and the baseline platform. Notice that we have only captured three of the five experimental factors at this point. The other two experimental factors, virtual machine instance type and availability zone status, are captured within the construct of the cloud platforms themselves. Four separate cloud platforms exist because we have two remaining factors that each have two levels, accounting for all possible combinations. This construct completes the full factorial experimental design.



				Cloud Platform 1		Cloud Platform 2		Cloud Platform 3		Cloud Platform 4	
Config #	Packet Size (Bytes)	Packet Arrival (Packets/Sec)	Protocol	VM Type	AV Zone	VM Type	AV Zone	VM Type	AV Zone	VM Type	AV Zone
1	Constant: 512	Constant: 1,000	TCP	M1.Medium	Same	M1.Large	Same	M1.Medium	Different	M1.Large	Different
2	Poisson: Mean 512	Uniform: Min=256,Max=4096	TCP	M1.Medium	Same	M1.Large	Same	M1.Medium	Different	M1.Large	Different
3	Uniform: Min=256,Max=4096	Poisson: Mean 512	TCP	M1.Medium	Same	M1.Large	Same	M1.Medium	Different	M1.Large	Different
4	Constant: 512	Uniform: Min=256,Max=4096	TCP	M1.Medium	Same	M1.Large	Same	M1.Medium	Different	M1.Large	Different
5	Poisson: Mean 512	Constant: 1,000	TCP	M1.Medium	Same	M1.Large	Same	M1.Medium	Different	M1.Large	Different
6	Uniform: Min=256,Max=4096	Uniform: Min=256,Max=4096	TCP	M1.Medium	Same	M1.Large	Same	M1.Medium	Different	M1.Large	Different
7	Constant: 512	Poisson: Mean 512	TCP	M1.Medium	Same	M1.Large	Same	M1.Medium	Different	M1.Large	Different
8	Poisson: Mean 512	Poisson: Mean 512	TCP	M1.Medium	Same	M1.Large	Same	M1.Medium	Different	M1.Large	Different
9	Uniform: Min=256,Max=4096	Constant: 1,000	TCP	M1.Medium	Same	M1.Large	Same	M1.Medium	Different	M1.Large	Different
10	Constant: 512	Constant: 1,000	UDP	M1.Medium	Same	M1.Large	Same	M1.Medium	Different	M1.Large	Different
11	Poisson: Mean 512	Uniform: Min=256,Max=4096	UDP	M1.Medium	Same	M1.Large	Same	M1.Medium	Different	M1.Large	Different
12	Uniform: Min=256,Max=4096	Poisson: Mean 512	UDP	M1.Medium	Same	M1.Large	Same	M1.Medium	Different	M1.Large	Different
13	Constant: 512	Uniform: Min=256,Max=4096	UDP	M1.Medium	Same	M1.Large	Same	M1.Medium	Different	M1.Large	Different
14	Poisson: Mean 512	Constant: 1,000	UDP	M1.Medium	Same	M1.Large	Same	M1.Medium	Different	M1.Large	Different
15	Uniform: Min=256,Max=4096	Uniform: Min=256,Max=4096	UDP	M1.Medium	Same	M1.Large	Same	M1.Medium	Different	M1.Large	Different
16	Constant: 512	Poisson: Mean 512	UDP	M1.Medium	Same	M1.Large	Same	M1.Medium	Different	M1.Large	Different
17	Poisson: Mean 512	Poisson: Mean 512	UDP	M1.Medium	Same	M1.Large	Same	M1.Medium	Different	M1.Large	Different
18	Uniform: Min=256,Max=4096	Constant: 1,000	UDP	M1.Medium	Same	M1.Large	Same	M1.Medium	Different	M1.Large	Different

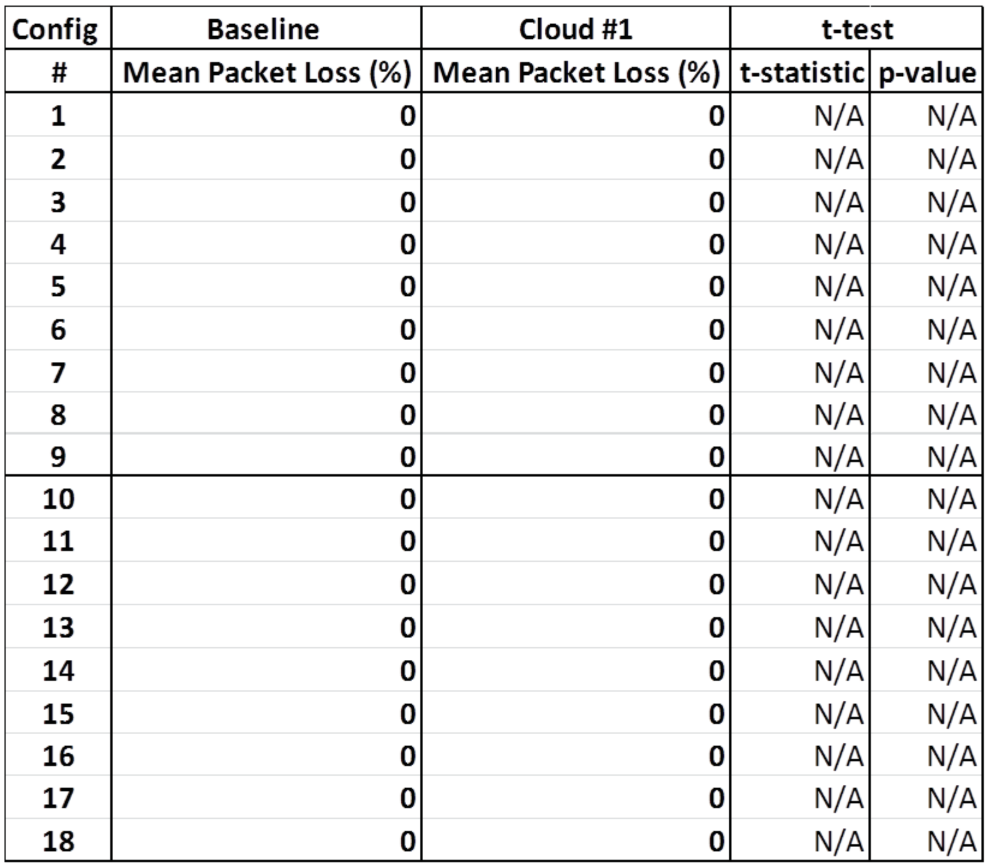
Figure 4.3: Traffic Workload Configurations

4.2.1 Packet Loss.

Figures 4.4, 4.5, 4.6 and 4.7 show the mean packet loss values for the baseline platform compared to cloud platforms 1, 2, 3 and 4 respectively. Cloud Platforms 2 and 4 are the only platforms that show any packet loss. These platforms are different than other cloud platforms in the sense that they both consist of M1.Large instance types. In both cases, t-statistic values with absolute values less than 2.0 coupled with p-values greater than 0.05 indicate a lack of statistical significance.

4.2.2 Delay.

Figures 4.8, 4.9, 4.10 and 4.11 show the mean delay values for all platforms under the 18 unique traffic workload configurations. Traffic workload configuration 14 is the only instance where baseline platform performance exceeds cloud platform performance. Results from 95 percent confidence intervals show that Cloud performance exceeds baseline performance for each of the remaining 17 configurations. In all cases, each cloud



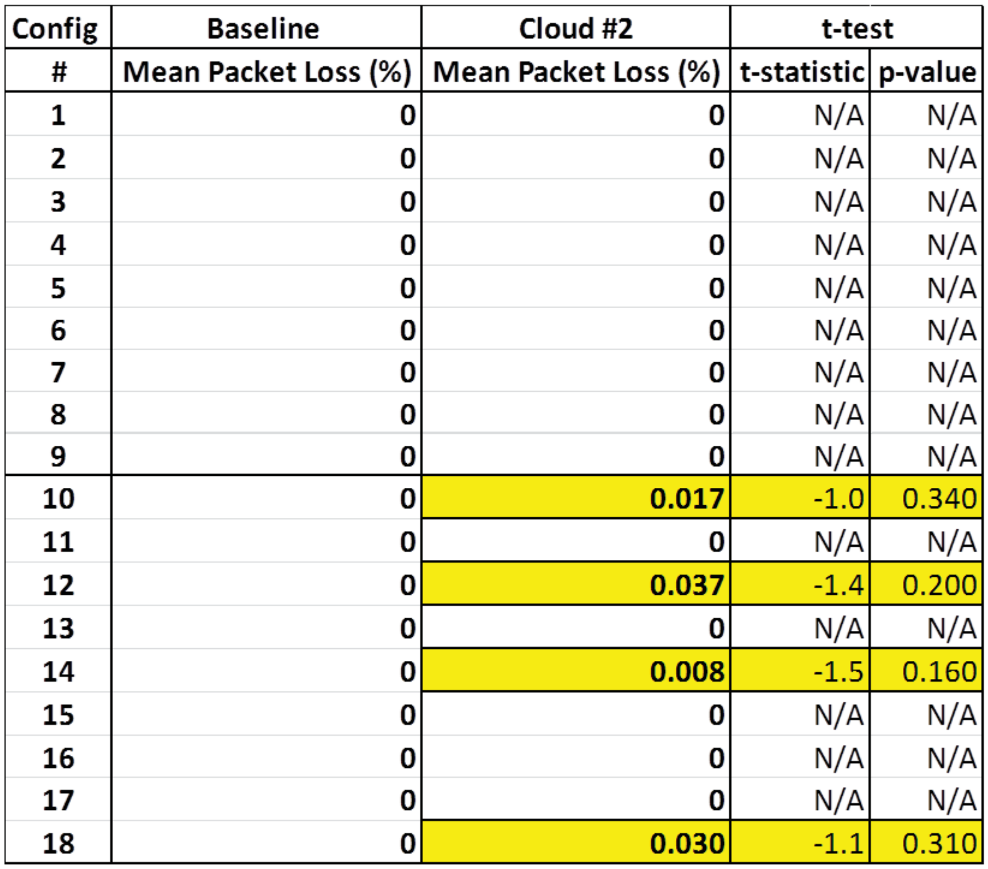
Config	Baseline	Cloud #1	t-test	
#	Mean Packet Loss (%)	Mean Packet Loss (%)	t-statistic	p-value
1	0	0	N/A	N/A
2	0	0	N/A	N/A
3	0	0	N/A	N/A
4	0	0	N/A	N/A
5	0	0	N/A	N/A
6	0	0	N/A	N/A
7	0	0	N/A	N/A
8	0	0	N/A	N/A
9	0	0	N/A	N/A
10	0	0	N/A	N/A
11	0	0	N/A	N/A
12	0	0	N/A	N/A
13	0	0	N/A	N/A
14	0	0	N/A	N/A
15	0	0	N/A	N/A
16	0	0	N/A	N/A
17	0	0	N/A	N/A
18	0	0	N/A	N/A

Figure 4.4: Packet Loss:Baseline vs Cloud Platform 1

platform shows delays less than 90 ms, meeting the acceptability threshold for a wide range of applications.

4.2.3 Jitter.

Figures 4.12, 4.13, 4.14 and 4.15 show the mean jitter values for all platforms under the 18 unique traffic workload configurations. This is the only metric where 95 percent confidence intervals show that the baseline platform's performance slightly exceeds that of three of the cloud platforms. However, each cloud platform produced jitter values less than 1 ms, which is acceptable for most applications.

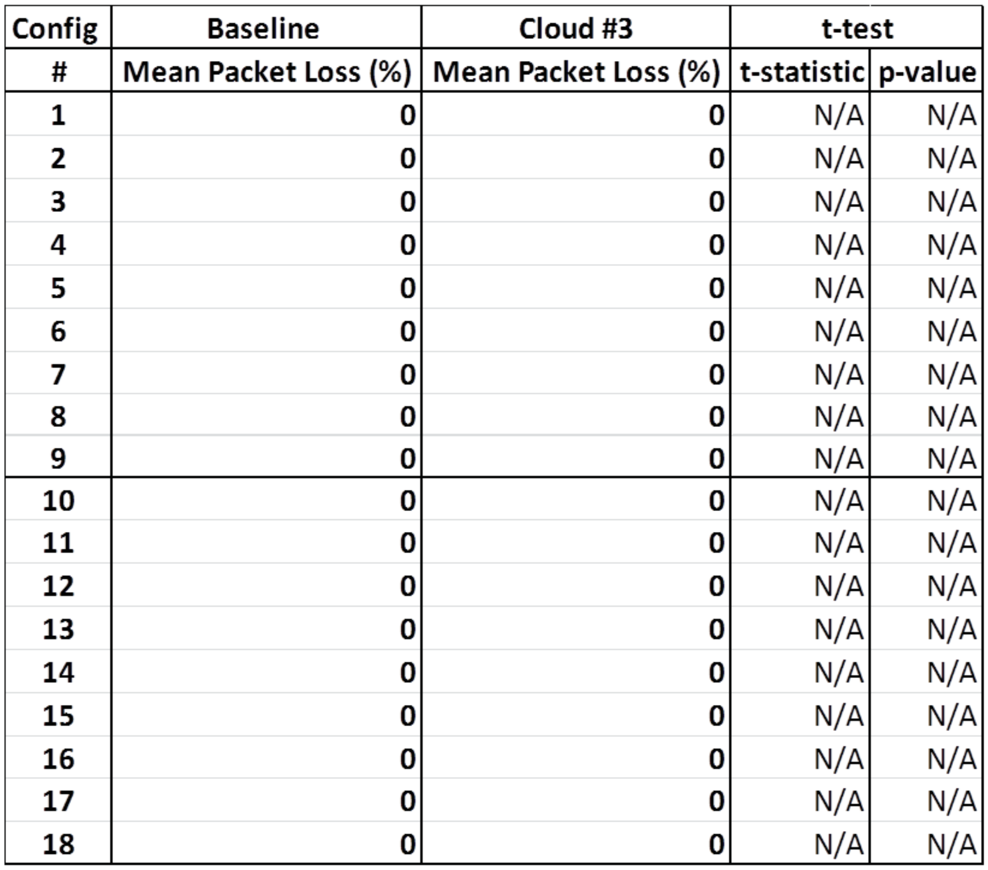


Config	Baseline	Cloud #2	t-test	
#	Mean Packet Loss (%)	Mean Packet Loss (%)	t-statistic	p-value
1	0	0	N/A	N/A
2	0	0	N/A	N/A
3	0	0	N/A	N/A
4	0	0	N/A	N/A
5	0	0	N/A	N/A
6	0	0	N/A	N/A
7	0	0	N/A	N/A
8	0	0	N/A	N/A
9	0	0	N/A	N/A
10	0	**0.017**	-1.0	0.340
11	0	0	N/A	N/A
12	0	**0.037**	-1.4	0.200
13	0	0	N/A	N/A
14	0	**0.008**	-1.5	0.160
15	0	0	N/A	N/A
16	0	0	N/A	N/A
17	0	0	N/A	N/A
18	0	**0.030**	-1.1	0.310

Figure 4.5: Packet Loss:Baseline vs Cloud Platform 2

4.2.4 Throughput.

Figures 4.16, 4.17, 4.18 and 4.19 show the mean throughput values for all platforms under the 18 unique traffic workload configurations. In every case, 95 percent confidence intervals show that cloud platforms produced higher throughput than the baseline platform. This is likely attributed to robustness of the cloud's underlying infrastructure, compared to running two virtual machines on a single personal computer. These values suggest that throughput performance for each cloud platform exceeds that of the baseline platform.



Config	Baseline	Cloud #3	t-test	
#	Mean Packet Loss (%)	Mean Packet Loss (%)	t-statistic	p-value
1	0	0	N/A	N/A
2	0	0	N/A	N/A
3	0	0	N/A	N/A
4	0	0	N/A	N/A
5	0	0	N/A	N/A
6	0	0	N/A	N/A
7	0	0	N/A	N/A
8	0	0	N/A	N/A
9	0	0	N/A	N/A
10	0	0	N/A	N/A
11	0	0	N/A	N/A
12	0	0	N/A	N/A
13	0	0	N/A	N/A
14	0	0	N/A	N/A
15	0	0	N/A	N/A
16	0	0	N/A	N/A
17	0	0	N/A	N/A
18	0	0	N/A	N/A

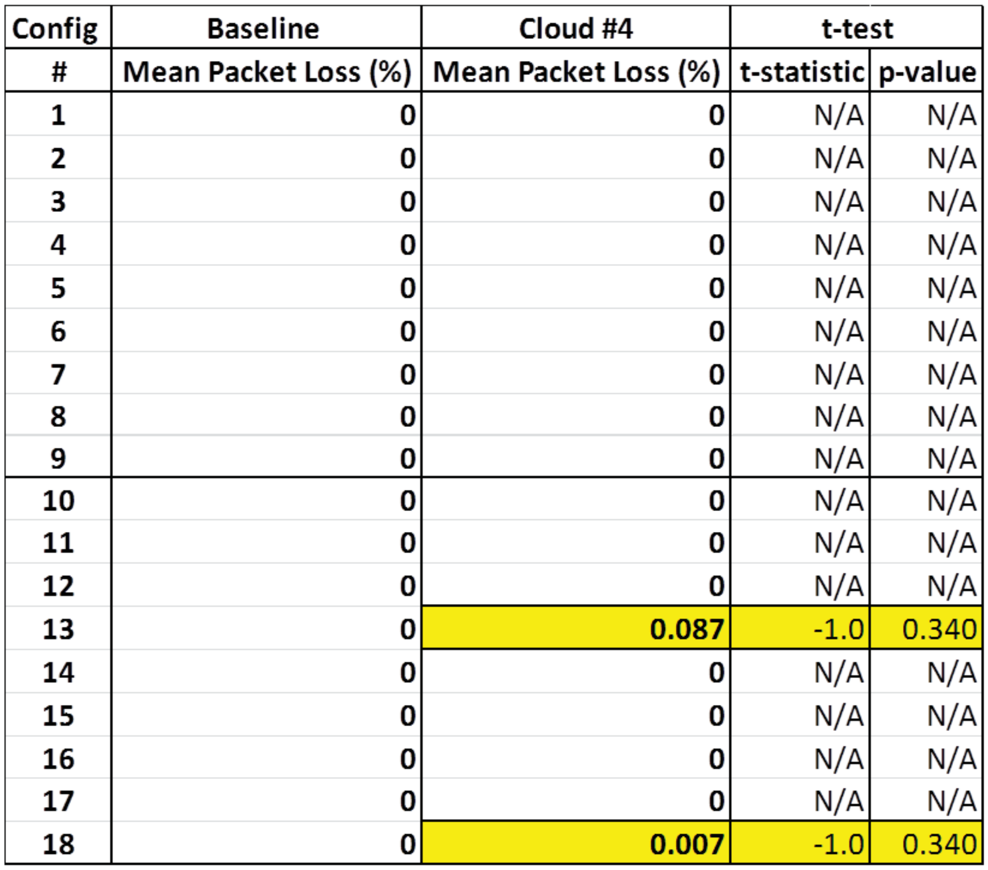
Figure 4.6: Packet Loss:Baseline vs Cloud Platform 3

4.3 Summary of Findings

4.3.1 Results Summary.

4.3.1.1 PING Data.

Regarding network latency as measured by the PING utility, 95 percent confidence intervals indicate that there is a difference between the baseline platform and each of the cloud platforms. This illustrates the effect of physical distance on network performance. PINGs travel a smaller distance when virtual machines are located within the same host, and round-trip time increases as physical distance increases. This is further illustrated by the higher latency observed in cloud platforms that are in different availability zones.



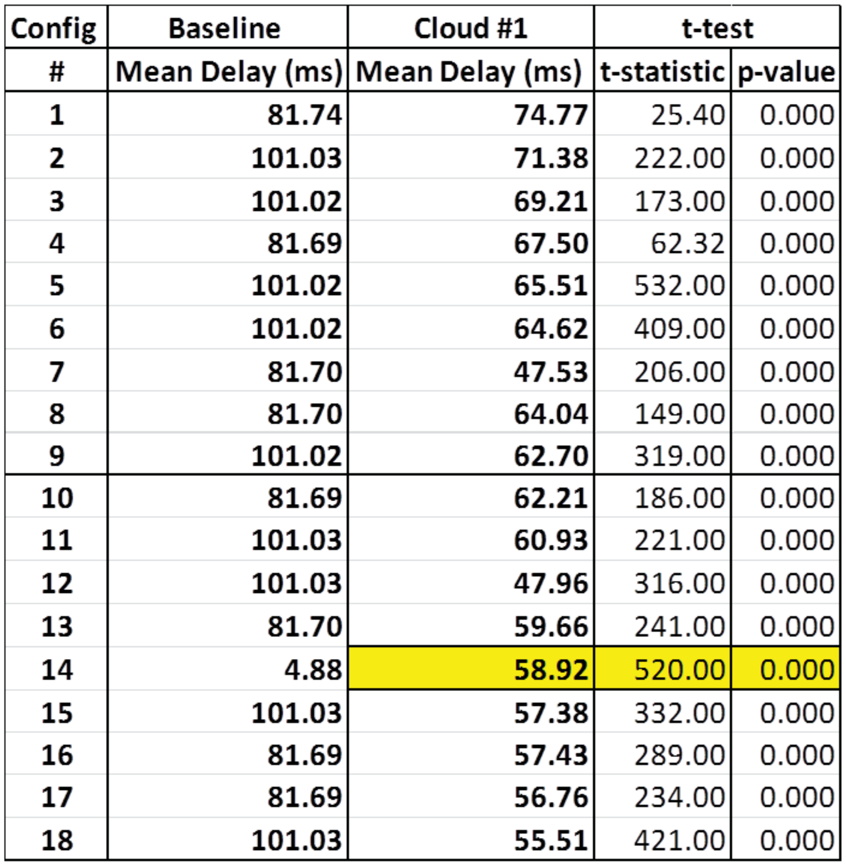
Config	Baseline	Cloud #4	t-test	
#	Mean Packet Loss (%)	Mean Packet Loss (%)	t-statistic	p-value
1	0	0	N/A	N/A
2	0	0	N/A	N/A
3	0	0	N/A	N/A
4	0	0	N/A	N/A
5	0	0	N/A	N/A
6	0	0	N/A	N/A
7	0	0	N/A	N/A
8	0	0	N/A	N/A
9	0	0	N/A	N/A
10	0	0	N/A	N/A
11	0	0	N/A	N/A
12	0	0	N/A	N/A
13	0	0.087	-1.0	0.340
14	0	0	N/A	N/A
15	0	0	N/A	N/A
16	0	0	N/A	N/A
17	0	0	N/A	N/A
18	0	0.007	-1.0	0.340

Figure 4.7: Packet Loss:Baseline vs Cloud Platform 4

However, latency values for all cloud platforms are small enough to be acceptable to a wide range of applications.

4.3.1.2 Packet Loss.

For packet loss, 95 percent confidence intervals indicate that there is no difference between the baseline platform and each of the cloud platforms. Since these confidence intervals cannot exclude a mean value of zero percent packet loss, they are acceptable for a wide range of applications. Therefore, network architectures constructed in the cloud are expected to display packet loss performance similar to that of traditional network architectures.

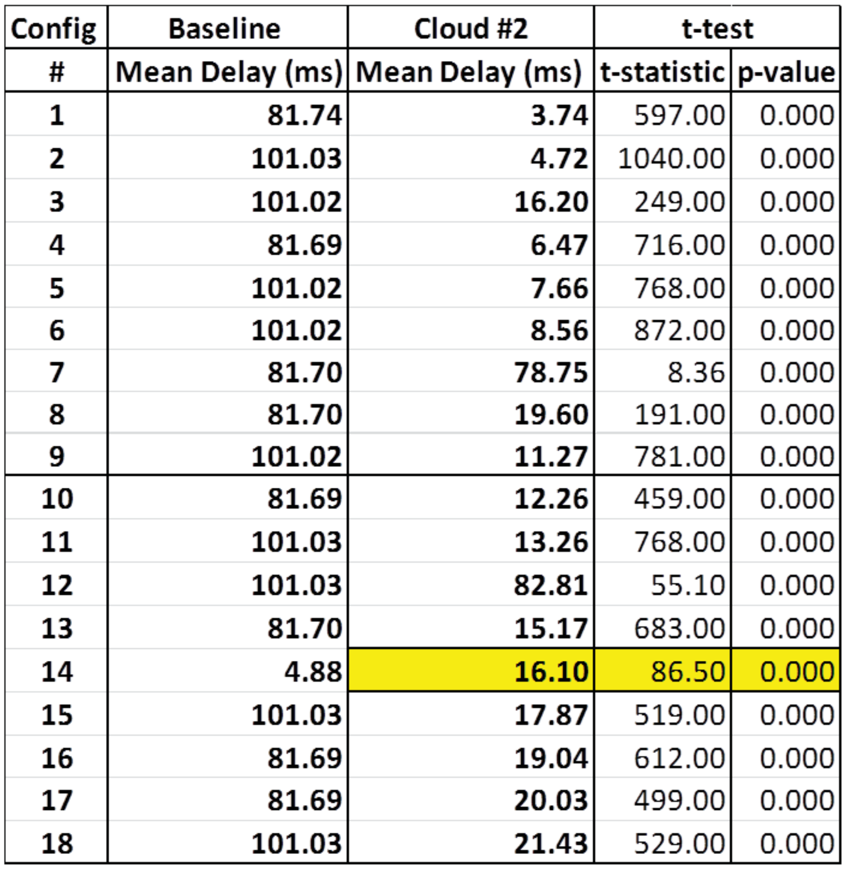


Config	Baseline	Cloud #1	t-test	
#	Mean Delay (ms)	Mean Delay (ms)	t-statistic	p-value
1	81.74	74.77	25.40	0.000
2	101.03	71.38	222.00	0.000
3	101.02	69.21	173.00	0.000
4	81.69	67.50	62.32	0.000
5	101.02	65.51	532.00	0.000
6	101.02	64.62	409.00	0.000
7	81.70	47.53	206.00	0.000
8	81.70	64.04	149.00	0.000
9	101.02	62.70	319.00	0.000
10	81.69	62.21	186.00	0.000
11	101.03	60.93	221.00	0.000
12	101.03	47.96	316.00	0.000
13	81.70	59.66	241.00	0.000
14	4.88	58.92	520.00	0.000
15	101.03	57.38	332.00	0.000
16	81.69	57.43	289.00	0.000
17	81.69	56.76	234.00	0.000
18	101.03	55.51	421.00	0.000

Figure 4.8: Delay:Baseline vs Cloud Platform 1

4.3.1.3 Delay.

Results show that 95 percent confidence intervals indicate a difference between the baseline platform and each of the cloud platforms for all 18 traffic workload configurations. Configuration number 14 is the only instance where baseline platform performance exceeds cloud performance. In each of the other 17 configurations, cloud performance exceeds baseline performance. All delay measurements observed in the cloud were less than 90 milliseconds, which is acceptable for a wide range of applications. Therefore, network architectures constructed in the cloud are expected to display delay performance similar to that of traditional network architectures.

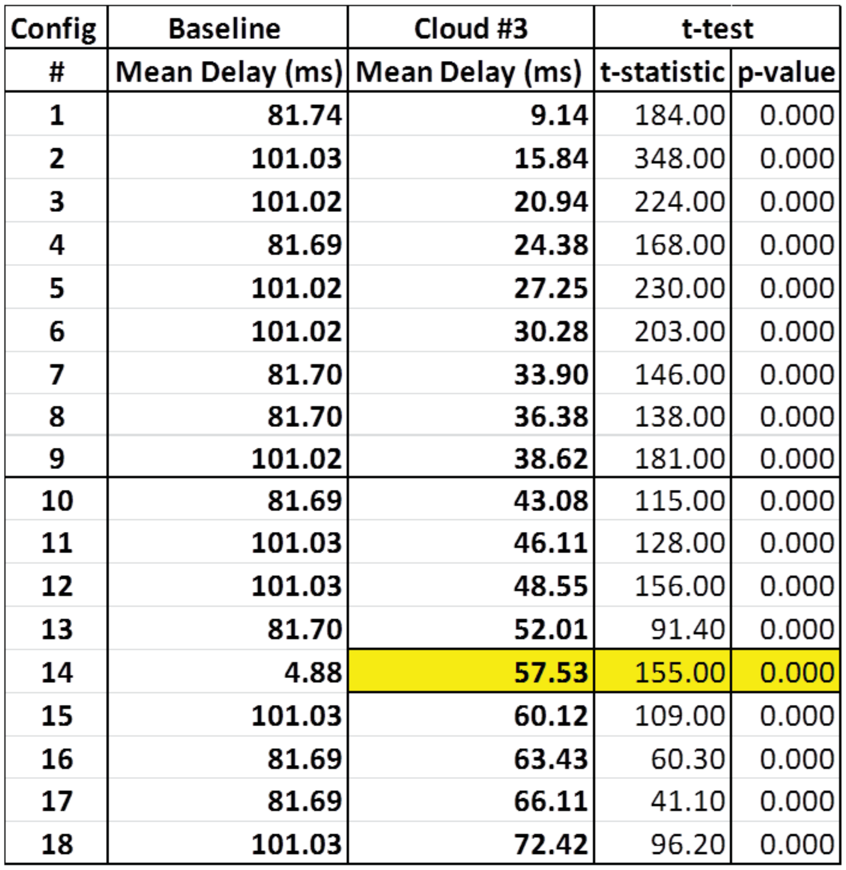


Config	Baseline	Cloud #2	t-test	
#	Mean Delay (ms)	Mean Delay (ms)	t-statistic	p-value
1	81.74	3.74	597.00	0.000
2	101.03	4.72	1040.00	0.000
3	101.02	16.20	249.00	0.000
4	81.69	6.47	716.00	0.000
5	101.02	7.66	768.00	0.000
6	101.02	8.56	872.00	0.000
7	81.70	78.75	8.36	0.000
8	81.70	19.60	191.00	0.000
9	101.02	11.27	781.00	0.000
10	81.69	12.26	459.00	0.000
11	101.03	13.26	768.00	0.000
12	101.03	82.81	55.10	0.000
13	81.70	15.17	683.00	0.000
14	4.88	16.10	86.50	0.000
15	101.03	17.87	519.00	0.000
16	81.69	19.04	612.00	0.000
17	81.69	20.03	499.00	0.000
18	101.03	21.43	529.00	0.000

Figure 4.9: Delay:Baseline vs Cloud Platform 2

4.3.1.4 Jitter.

Results for jitter performance are not as conclusive as other metrics. For example, out of 18 traffic workload configurations, 95 percent confidence intervals show better performance for Cloud Platform 1 in five instances, better baseline performance in four instances, and no statistical difference in the other nine. All other cloud platforms show slightly more instances where baseline performance exceeds cloud performance. In all cases, cloud platforms show jitter values that are less than one millisecond, which is acceptable for a wide range of applications. Therefore, network architectures constructed in the cloud are expected to display jitter performance similar to that of traditional network architectures.



Config	Baseline	Cloud #3	t-test	
#	Mean Delay (ms)	Mean Delay (ms)	t-statistic	p-value
1	**81.74**	**9.14**	184.00	0.000
2	**101.03**	**15.84**	348.00	0.000
3	**101.02**	**20.94**	224.00	0.000
4	**81.69**	**24.38**	168.00	0.000
5	**101.02**	**27.25**	230.00	0.000
6	**101.02**	**30.28**	203.00	0.000
7	**81.70**	**33.90**	146.00	0.000
8	**81.70**	**36.38**	138.00	0.000
9	**101.02**	**38.62**	181.00	0.000
10	**81.69**	**43.08**	115.00	0.000
11	**101.03**	**46.11**	128.00	0.000
12	**101.03**	**48.55**	156.00	0.000
13	**81.70**	**52.01**	91.40	0.000
14	**4.88**	**57.53**	155.00	0.000
15	**101.03**	**60.12**	109.00	0.000
16	**81.69**	**63.43**	60.30	0.000
17	**81.69**	**66.11**	41.10	0.000
18	**101.03**	**72.42**	96.20	0.000

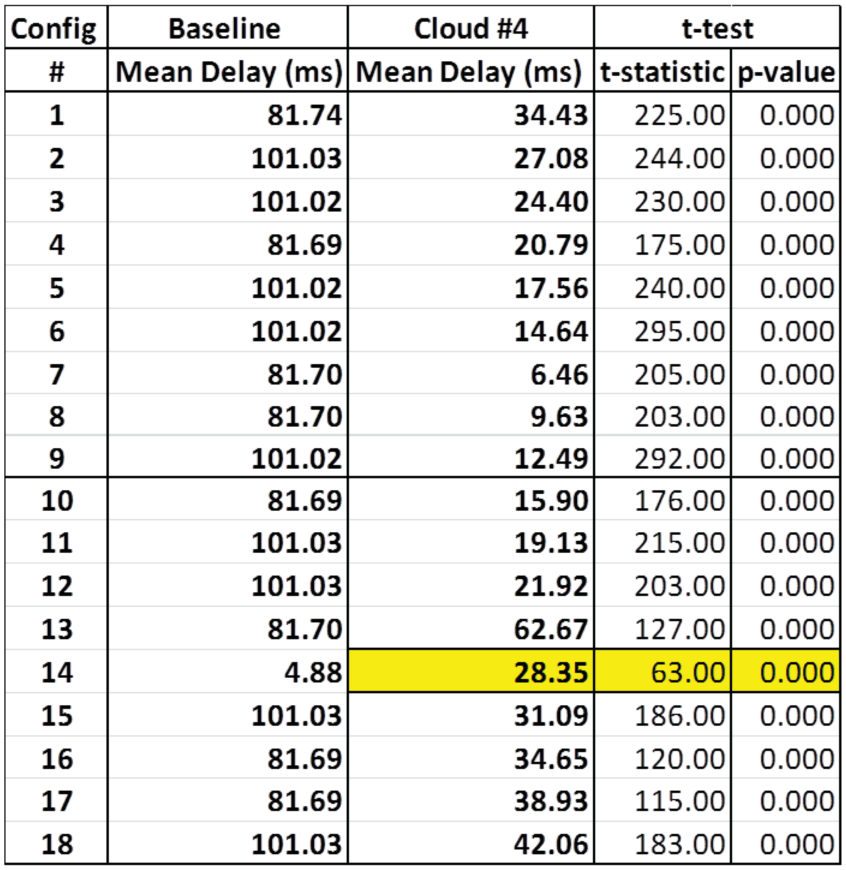
Figure 4.10: Delay:Baseline vs Cloud Platform 3

4.3.1.5 Throughput.

Regarding throughput, 95 percent confidence intervals show that all cloud platforms achieved higher throughput than the baseline platform under the statistically equivalent workload in all 18 configurations. This is likely attributed to the larger amount of resources available in the cloud to power the virtual machines and pass traffic between them. Therefore, there is no evidence that network architectures constructed in the cloud would not mimic the performance of networks outside of the cloud.

4.3.2 Scope of Inference.

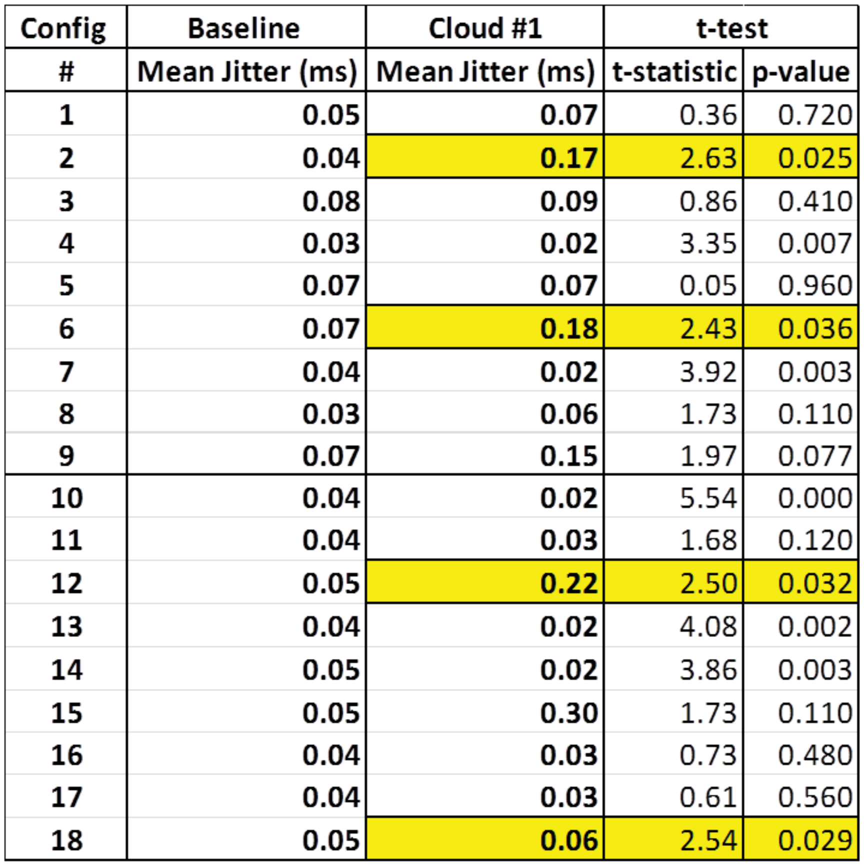
This is a random experiment, therefore a causal link between baseline platform and cloud platform performance can be validly inferred. Additionally, the Amazon region is



Config	Baseline	Cloud #4	t-test	
#	Mean Delay (ms)	Mean Delay (ms)	t-statistic	p-value
1	**81.74**	**34.43**	225.00	0.000
2	**101.03**	**27.08**	244.00	0.000
3	**101.02**	**24.40**	230.00	0.000
4	**81.69**	**20.79**	175.00	0.000
5	**101.02**	**17.56**	240.00	0.000
6	**101.02**	**14.64**	295.00	0.000
7	**81.70**	**6.46**	205.00	0.000
8	**81.70**	**9.63**	203.00	0.000
9	**101.02**	**12.49**	292.00	0.000
10	**81.69**	**15.90**	176.00	0.000
11	**101.03**	**19.13**	215.00	0.000
12	**101.03**	**21.92**	203.00	0.000
13	**81.70**	**62.67**	127.00	0.000
14	**4.88**	**28.35**	63.00	0.000
15	**101.03**	**31.09**	186.00	0.000
16	**81.69**	**34.65**	120.00	0.000
17	**81.69**	**38.93**	115.00	0.000
18	**101.03**	**42.06**	183.00	0.000

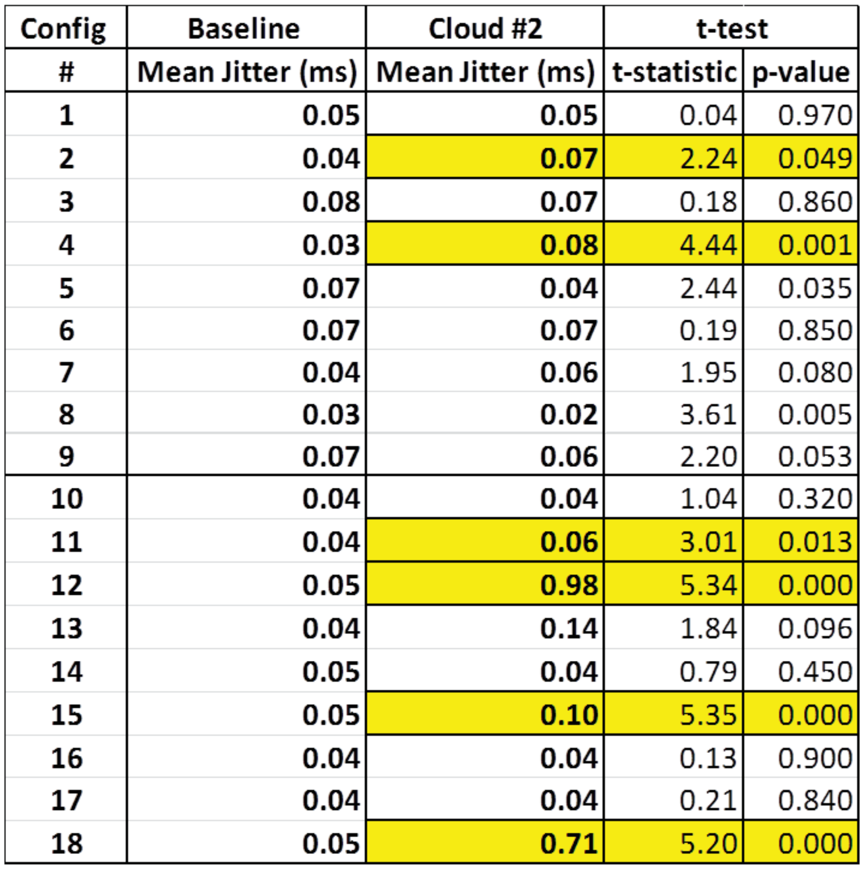
Figure 4.11: Delay:Baseline vs Cloud Platform 4

selected randomly. Therefore, inference can be made to apply these results to other Amazon regions.



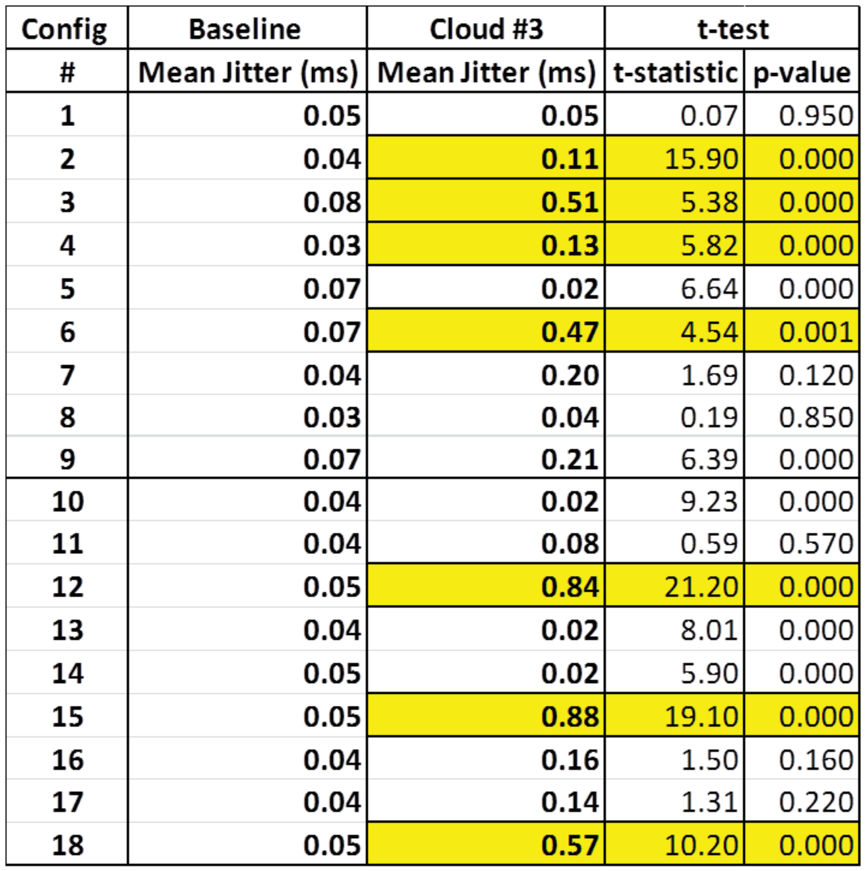
Config	Baseline	Cloud #1	t-test	
#	Mean Jitter (ms)	Mean Jitter (ms)	t-statistic	p-value
1	**0.05**	**0.07**	0.36	0.720
2	**0.04**	**0.17**	2.63	0.025
3	**0.08**	**0.09**	0.86	0.410
4	**0.03**	**0.02**	3.35	0.007
5	**0.07**	**0.07**	0.05	0.960
6	**0.07**	**0.18**	2.43	0.036
7	**0.04**	**0.02**	3.92	0.003
8	**0.03**	**0.06**	1.73	0.110
9	**0.07**	**0.15**	1.97	0.077
10	**0.04**	**0.02**	5.54	0.000
11	**0.04**	**0.03**	1.68	0.120
12	**0.05**	**0.22**	2.50	0.032
13	**0.04**	**0.02**	4.08	0.002
14	**0.05**	**0.02**	3.86	0.003
15	**0.05**	**0.30**	1.73	0.110
16	**0.04**	**0.03**	0.73	0.480
17	**0.04**	**0.03**	0.61	0.560
18	**0.05**	**0.06**	2.54	0.029

Figure 4.12: Jitter:Baseline vs Cloud Platform 1



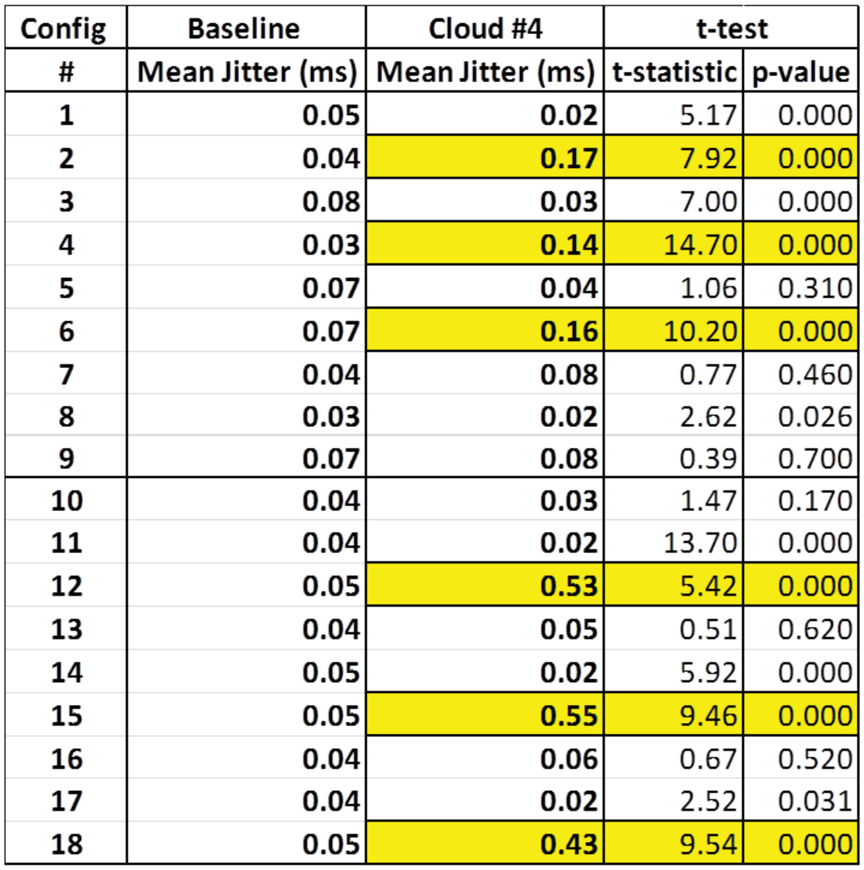
Config	Baseline	Cloud #2	t-test	
#	Mean Jitter (ms)	Mean Jitter (ms)	t-statistic	p-value
1	0.05	0.05	0.04	0.970
2	0.04	0.07	2.24	0.049
3	0.08	0.07	0.18	0.860
4	0.03	0.08	4.44	0.001
5	0.07	0.04	2.44	0.035
6	0.07	0.07	0.19	0.850
7	0.04	0.06	1.95	0.080
8	0.03	0.02	3.61	0.005
9	0.07	0.06	2.20	0.053
10	0.04	0.04	1.04	0.320
11	0.04	0.06	3.01	0.013
12	0.05	0.98	5.34	0.000
13	0.04	0.14	1.84	0.096
14	0.05	0.04	0.79	0.450
15	0.05	0.10	5.35	0.000
16	0.04	0.04	0.13	0.900
17	0.04	0.04	0.21	0.840
18	0.05	0.71	5.20	0.000

Figure 4.13: Jitter:Baseline vs Cloud Platform 2



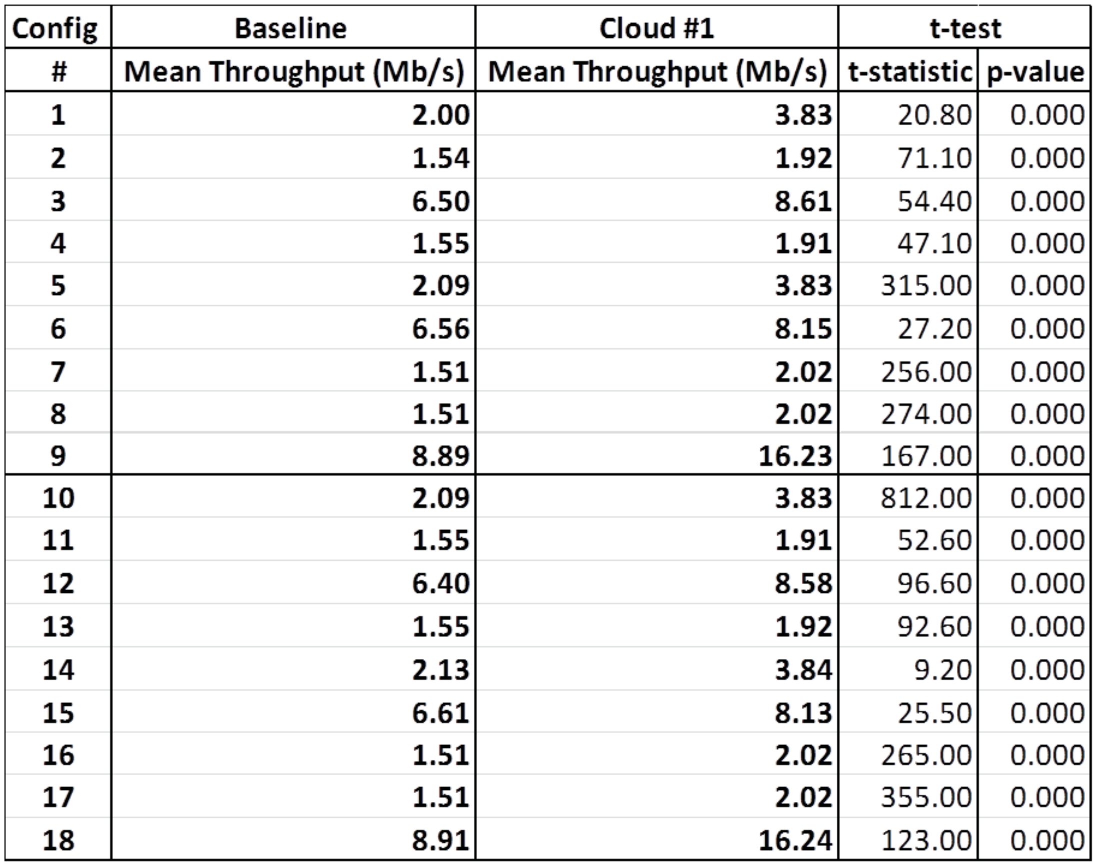
Config	Baseline	Cloud #3	t-test	
#	Mean Jitter (ms)	Mean Jitter (ms)	t-statistic	p-value
1	**0.05**	**0.05**	0.07	0.950
2	**0.04**	**0.11**	15.90	0.000
3	**0.08**	**0.51**	5.38	0.000
4	**0.03**	**0.13**	5.82	0.000
5	**0.07**	**0.02**	6.64	0.000
6	**0.07**	**0.47**	4.54	0.001
7	**0.04**	**0.20**	1.69	0.120
8	**0.03**	**0.04**	0.19	0.850
9	**0.07**	**0.21**	6.39	0.000
10	**0.04**	**0.02**	9.23	0.000
11	**0.04**	**0.08**	0.59	0.570
12	**0.05**	**0.84**	21.20	0.000
13	**0.04**	**0.02**	8.01	0.000
14	**0.05**	**0.02**	5.90	0.000
15	**0.05**	**0.88**	19.10	0.000
16	**0.04**	**0.16**	1.50	0.160
17	**0.04**	**0.14**	1.31	0.220
18	**0.05**	**0.57**	10.20	0.000

Figure 4.14: Jitter:Baseline vs Cloud Platform 3



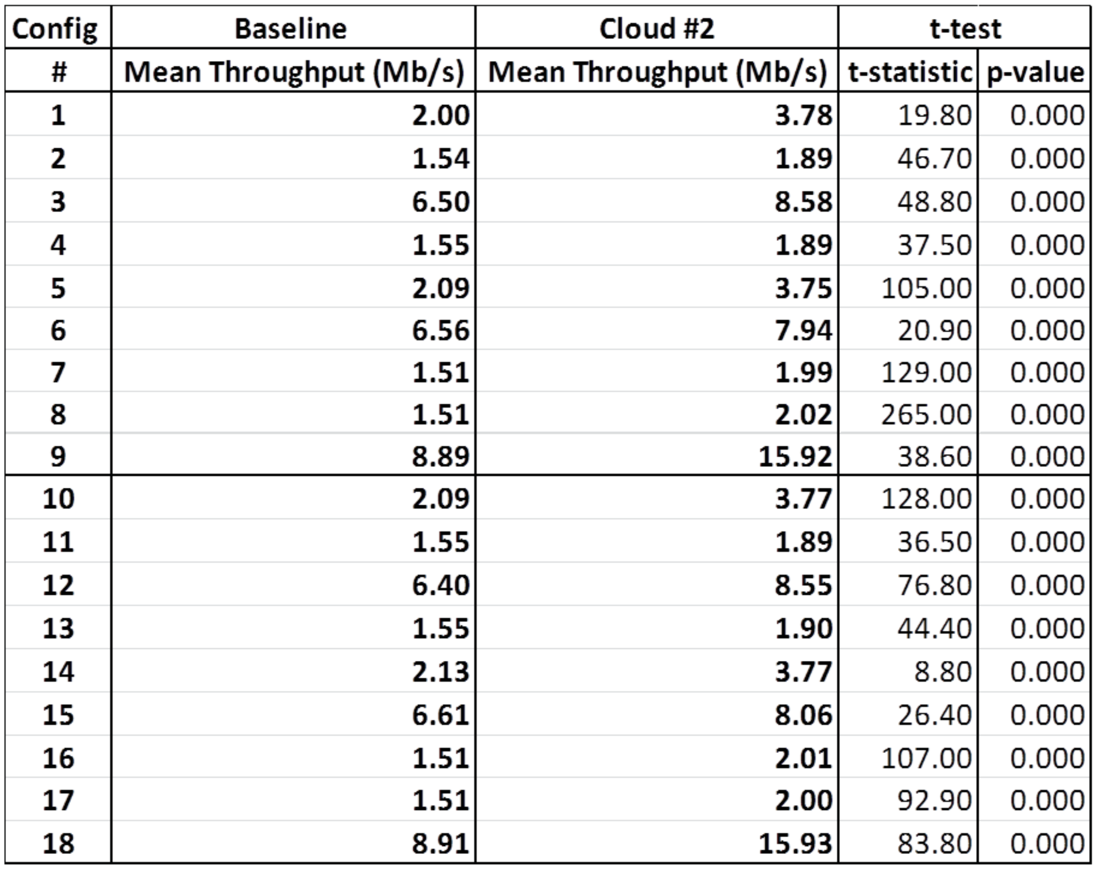
Config	Baseline	Cloud #4	t-test	
#	Mean Jitter (ms)	Mean Jitter (ms)	t-statistic	p-value
1	0.05	0.02	5.17	0.000
2	0.04	0.17	7.92	0.000
3	0.08	0.03	7.00	0.000
4	0.03	0.14	14.70	0.000
5	0.07	0.04	1.06	0.310
6	0.07	0.16	10.20	0.000
7	0.04	0.08	0.77	0.460
8	0.03	0.02	2.62	0.026
9	0.07	0.08	0.39	0.700
10	0.04	0.03	1.47	0.170
11	0.04	0.02	13.70	0.000
12	0.05	0.53	5.42	0.000
13	0.04	0.05	0.51	0.620
14	0.05	0.02	5.92	0.000
15	0.05	0.55	9.46	0.000
16	0.04	0.06	0.67	0.520
17	0.04	0.02	2.52	0.031
18	0.05	0.43	9.54	0.000

Figure 4.15: Jitter:Baseline vs Cloud Platform 4



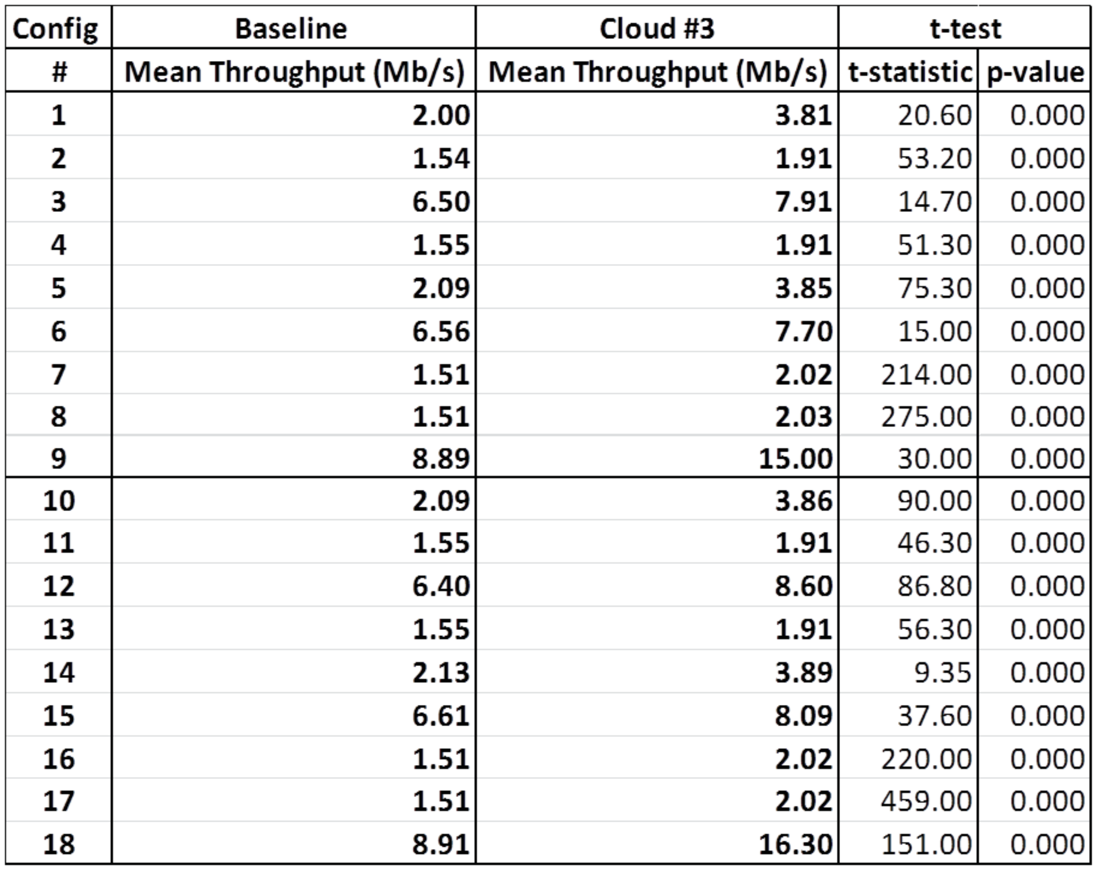
Config	Baseline	Cloud #1	t-test	
#	Mean Throughput (Mb/s)	Mean Throughput (Mb/s)	t-statistic	p-value
1	**2.00**	**3.83**	20.80	0.000
2	**1.54**	**1.92**	71.10	0.000
3	**6.50**	**8.61**	54.40	0.000
4	**1.55**	**1.91**	47.10	0.000
5	**2.09**	**3.83**	315.00	0.000
6	**6.56**	**8.15**	27.20	0.000
7	**1.51**	**2.02**	256.00	0.000
8	**1.51**	**2.02**	274.00	0.000
9	**8.89**	**16.23**	167.00	0.000
10	**2.09**	**3.83**	812.00	0.000
11	**1.55**	**1.91**	52.60	0.000
12	**6.40**	**8.58**	96.60	0.000
13	**1.55**	**1.92**	92.60	0.000
14	**2.13**	**3.84**	9.20	0.000
15	**6.61**	**8.13**	25.50	0.000
16	**1.51**	**2.02**	265.00	0.000
17	**1.51**	**2.02**	355.00	0.000
18	**8.91**	**16.24**	123.00	0.000

Figure 4.16: Throughput:Baseline vs Cloud Platform 1



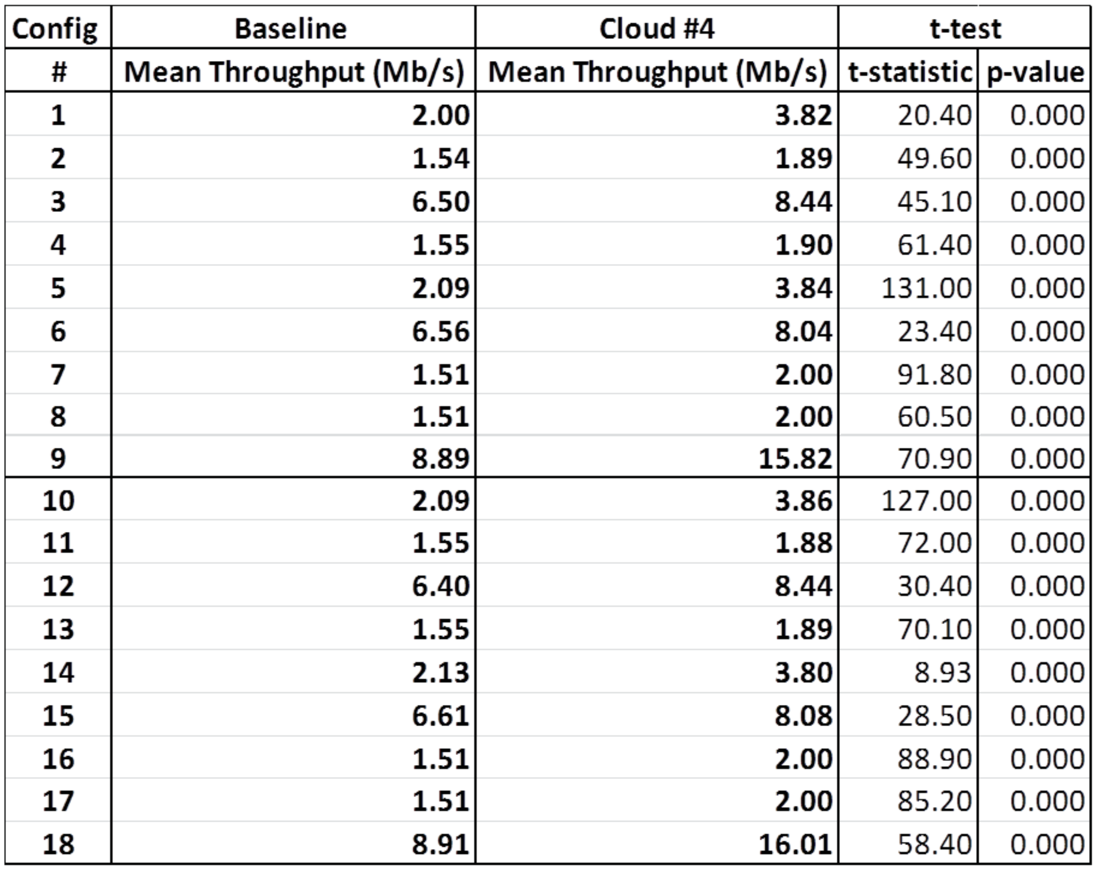
Config	Baseline	Cloud #2	t-test	
#	Mean Throughput (Mb/s)	Mean Throughput (Mb/s)	t-statistic	p-value
1	**2.00**	**3.78**	19.80	0.000
2	**1.54**	**1.89**	46.70	0.000
3	**6.50**	**8.58**	48.80	0.000
4	**1.55**	**1.89**	37.50	0.000
5	**2.09**	**3.75**	105.00	0.000
6	**6.56**	**7.94**	20.90	0.000
7	**1.51**	**1.99**	129.00	0.000
8	**1.51**	**2.02**	265.00	0.000
9	**8.89**	**15.92**	38.60	0.000
10	**2.09**	**3.77**	128.00	0.000
11	**1.55**	**1.89**	36.50	0.000
12	**6.40**	**8.55**	76.80	0.000
13	**1.55**	**1.90**	44.40	0.000
14	**2.13**	**3.77**	8.80	0.000
15	**6.61**	**8.06**	26.40	0.000
16	**1.51**	**2.01**	107.00	0.000
17	**1.51**	**2.00**	92.90	0.000
18	**8.91**	**15.93**	83.80	0.000

Figure 4.17: Throughput:Baseline vs Cloud Platform 2



Config	Baseline	Cloud #3	t-test	
#	Mean Throughput (Mb/s)	Mean Throughput (Mb/s)	t-statistic	p-value
1	2.00	3.81	20.60	0.000
2	1.54	1.91	53.20	0.000
3	6.50	7.91	14.70	0.000
4	1.55	1.91	51.30	0.000
5	2.09	3.85	75.30	0.000
6	6.56	7.70	15.00	0.000
7	1.51	2.02	214.00	0.000
8	1.51	2.03	275.00	0.000
9	8.89	15.00	30.00	0.000
10	2.09	3.86	90.00	0.000
11	1.55	1.91	46.30	0.000
12	6.40	8.60	86.80	0.000
13	1.55	1.91	56.30	0.000
14	2.13	3.89	9.35	0.000
15	6.61	8.09	37.60	0.000
16	1.51	2.02	220.00	0.000
17	1.51	2.02	459.00	0.000
18	8.91	16.30	151.00	0.000

Figure 4.18: Throughput:Baseline vs Cloud Platform 3



Config	Baseline	Cloud #4	t-test	
#	Mean Throughput (Mb/s)	Mean Throughput (Mb/s)	t-statistic	p-value
1	2.00	3.82	20.40	0.000
2	1.54	1.89	49.60	0.000
3	6.50	8.44	45.10	0.000
4	1.55	1.90	61.40	0.000
5	2.09	3.84	131.00	0.000
6	6.56	8.04	23.40	0.000
7	1.51	2.00	91.80	0.000
8	1.51	2.00	60.50	0.000
9	8.89	15.82	70.90	0.000
10	2.09	3.86	127.00	0.000
11	1.55	1.88	72.00	0.000
12	6.40	8.44	30.40	0.000
13	1.55	1.89	70.10	0.000
14	2.13	3.80	8.93	0.000
15	6.61	8.08	28.50	0.000
16	1.51	2.00	88.90	0.000
17	1.51	2.00	85.20	0.000
18	8.91	16.01	58.40	0.000

Figure 4.19: Throughput:Baseline vs Cloud Platform 4

V. Distributed Network Application in the Cloud Case Study

This chapter describes the involvement of this research in a case study to build a distributed network application in the cloud. The discussion begins with a background introduction to the case study, before introducing ZeroMQ, the tool used to accomplish the task. It then describes the particular messaging framework within ZeroMQ used to modify the distributed network application to operate in a cloud environment before presenting results.

5.1 Introduction to the Case Study

Involvement in the Distributed Network Application in the Cloud case study is part of a larger data analysis effort. Boeing, in conjunction with Morgan State University, is currently in the process of conducting a data analysis experiment on flight simulation data. They aim to receive a capture of real-time flight simulation data that includes time-stamped position updates for each individual node as well as each node's perception about the location of every other node. This allows an analysis of the difference between *truth data*, which is the location that each node reports as its true location, and each node's calculated world picture after piecing together updates from all other nodes. Due to network latency, dropped update packets and other factors common to distributed applications, differences are likely to be present.

One of the issues faced by the data analysis effort is generating and receiving the flight data to be analyzed. In lieu of using more expensive solutions, such as having real air planes or building a flight simulation test range to generate position data to transfer to their networks, the data analysis team chose to leverage the public cloud as a testbed for generating and transferring the data. This research contributes to the case study by building the distributed application framework in the cloud, and extracting the data from the cloud to the Boeing and Morgan State networks for analysis.

5.2 ZeroMQ

The Open Extensible Architecture for the Analysis and Generation of Linked Simulations (OpenEaagles) is a simulation framework developed and maintained by the U.S. Air Force to support a multitude of simulation activities [27]. OpenEaagles has been used by the U.S. Air Force to transfer real-time flight simulation data on a number of projects. Currently, OpenEaagles uses UDP broacast packets to distribute information between network nodes. Due to Amazon's restriction on broadcast traffic, the application demands modification in order to operate in the cloud. An alternative option is needed that accomplishes the task of sending position updates to all nodes without sending broadcast packets. The framework known as ZeroMQ meets this need. ZeroMQ uses sockets to create a messaging framework that can transport any type of data between other ZeroMQ nodes [28]. ZeroMQ is an open source framework that supports multiple operating system platforms and multiple programming languages. Additionally, it features sockets that support unicast and multicast transports. These sockets express certain message patterns that are not necessarily one-to-one [29]. These message patterns are what allows ZeroMQ to send unicast packets and have them distributed in a broadcast or multicast fashion, thus eliminating the need to actually send broadcast or multicast packets.

5.3 Publish-Subscribe Messaging Framework

ZeroMQ has several messaging patterns available depending on user needs. The Publish-Subscribe pattern aims to create highly scalable group messaging by enabling users to send large volumes of data rapidly to many recipients. One of the ways that ZeroMQ achieves that scalability is by not having recipients talk back to senders. In other words, subscribers do not connect to the publisher at all, but rather a multicast group on a network switch, to which the publisher sends its messages. Publish-Subscribe is like a radio broadcast; the subscriber misses everything that happens prior to subscribing, and the amount of information received depends on the quality of reception [28]. While removing back-chatter simplifies message flow, it also removes the ability to coordinate between senders and receivers. This dynamic creates the following challenges [29]:

1. Publishers cannot detect whether subscribers are successfully connected, both on initial connections and re-connections after network failures.
2. Subscribers cannot coordinate with publishers regarding the message sending rate, causing subscribers to either keep up or lose messages.
3. Publishers cannot detect when subscribers have disappeared due to complications such as processes crashing, network outages, etc.
4. If subscribers join late or drop off, they miss messages sent by the publisher while not online.
5. If subscribers fetch messages too slowly, queues can build up and overflow.
6. If subscribers crash and restart, they lose the data that was already received.

The challenges listed above make this pattern unusable for applications that demand reliable multicast. However, some real-time distributed applications can tolerate almost reliable multicast due to their nature. For example, if a real-time position update is lost by the network, then re-transmitted, the position information is likely too outdated to be useful to the receiver once it finally arrives. It is better for the receiver in that case to wait for the next position update, which will follow shortly afterwards, due to the real-time nature of the updates.

5.4 Proof of Concept

Rather than immediately modifying the OpenEaagles source code to use ZeroMQ as a method of transport, a proof of concept experiment answers the functionality question with minimal programming requirements. The proof of concept consists of the following elements:

1. Virtual Machine Instances in the Cloud - Three virtual machine instances in the Amazon cloud serves as a publisher, subscriber, and proxy for the experiment.

2. Host on Boeing Network - This experiment uses a machine on the Boeing network as the host that receives data from the cloud.

3. Execution Programs - In lieu of writing a program that simulates actual flight position data, the experiment uses a simple weather update server construct using C++ programs. The publisher program generates random weather data that includes temperatures for various zipcodes in a continuous loop. The subscriber program chooses a zipcode and the number of weather updates to process for that zipcode. After receiving the required amount of weather updates from the publisher, the subscriber calculates the average temperature for that zipcode and displays the result to the user. Since the publisher and subscriber are not directly connected, a proxy program is needed. The proxy program subscribes to everything from the publisher on one socket, and publishes the same data on another socket. This allows subscribers to to the proxy for information. The benefit to this construct is that a single proxy can perform this function for multiple publishers and subscribers, without any of the subscribers needing to have knowledge of any of the publishers. Only the proxy's publish and subscribe socket addresses need to be known.

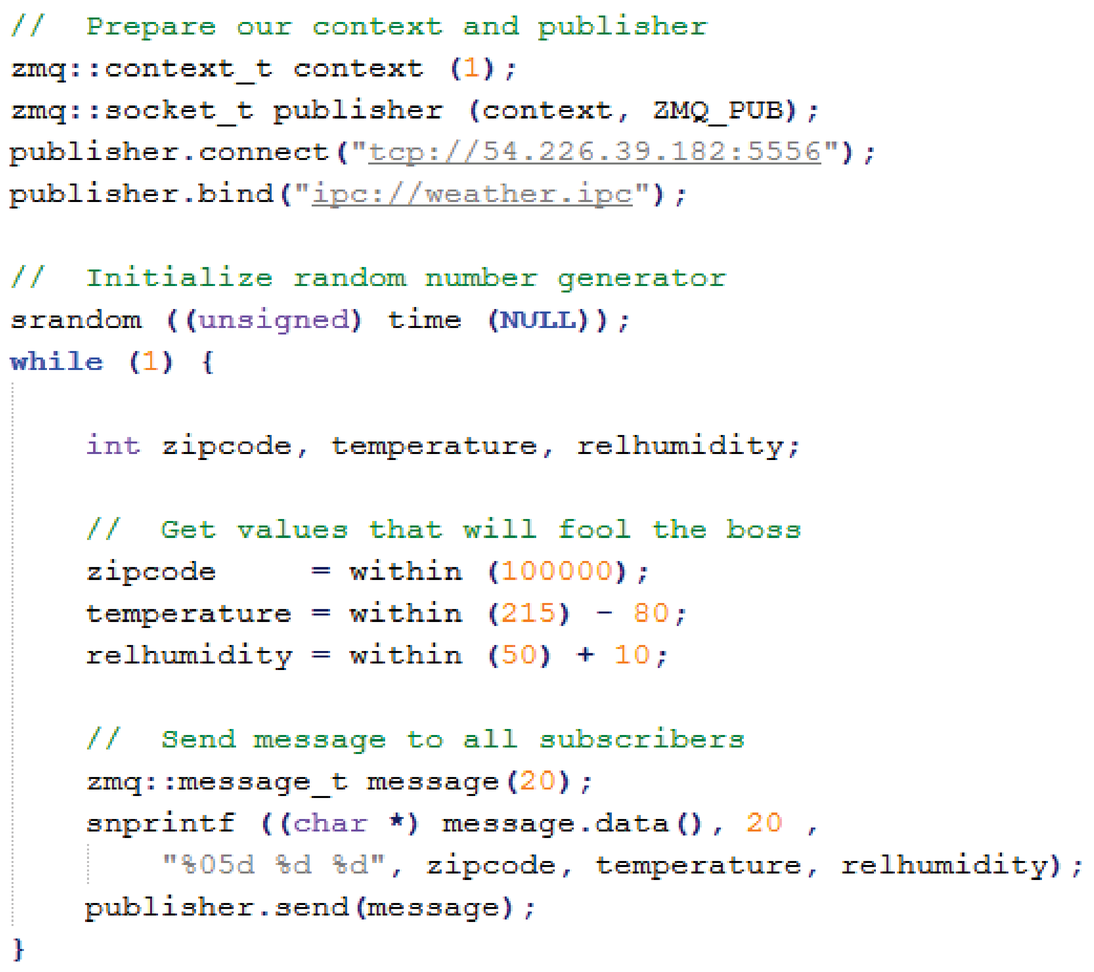
5.4.1 Publisher Program.

Figure 5.1 shows the applicable portion of the weather update publisher program. The publisher begins by preparing the context. The public IP address of the proxy machine in the Amazon cloud is 54.226.39.182. The publisher reaches out to this machine and establishes a connection to the proxy on port 5556. Although the publisher is establishing the connection, this socket is used to allow the proxy to subscribe to the publisher. This is made possible because under the ZeroMQ infrastructure, the direction in which a connection is established is not connected to the direction of traffic flow. For example, in typical web server communication, the web client establishes a connection to the server before traffic begins to flow. Web servers do not initiate connections to clients in order to send them data. ZeroMQ removes this restriction, allowing the server to connect to the proxy, while clients also connect to the same proxy. The proxy relays traffic back and forth. Therefore, as long as all entities can connect to the proxy, which has a public IP address, traffic can flow from any publisher to any subscriber.

After establishing the connection to the proxy, the publisher needs to actually publish data for subscribers. The publisher does this by binding weather.ipc in the next line of code. The publisher then uses a random number generator to generate zipcode, temperature and relative humidity data, and prepares messages to send to subscribers. This construct simulates a broadcast or multicast messaging environment because although all data gets published, subscribers choose the data to which to subscribe, which can be all or a subset of the data.

5.4.2 Subscriber Program.

Figure 5.2 shows the applicable portion of the weather update subscriber program. The program first connects to the same proxy address of 54.226.39.182, but on port 5559 rather than port 5556. This establishes a subscription to data relayed from the publisher via the proxy. The program defaults to collecting data for New York City zipcode 10001.



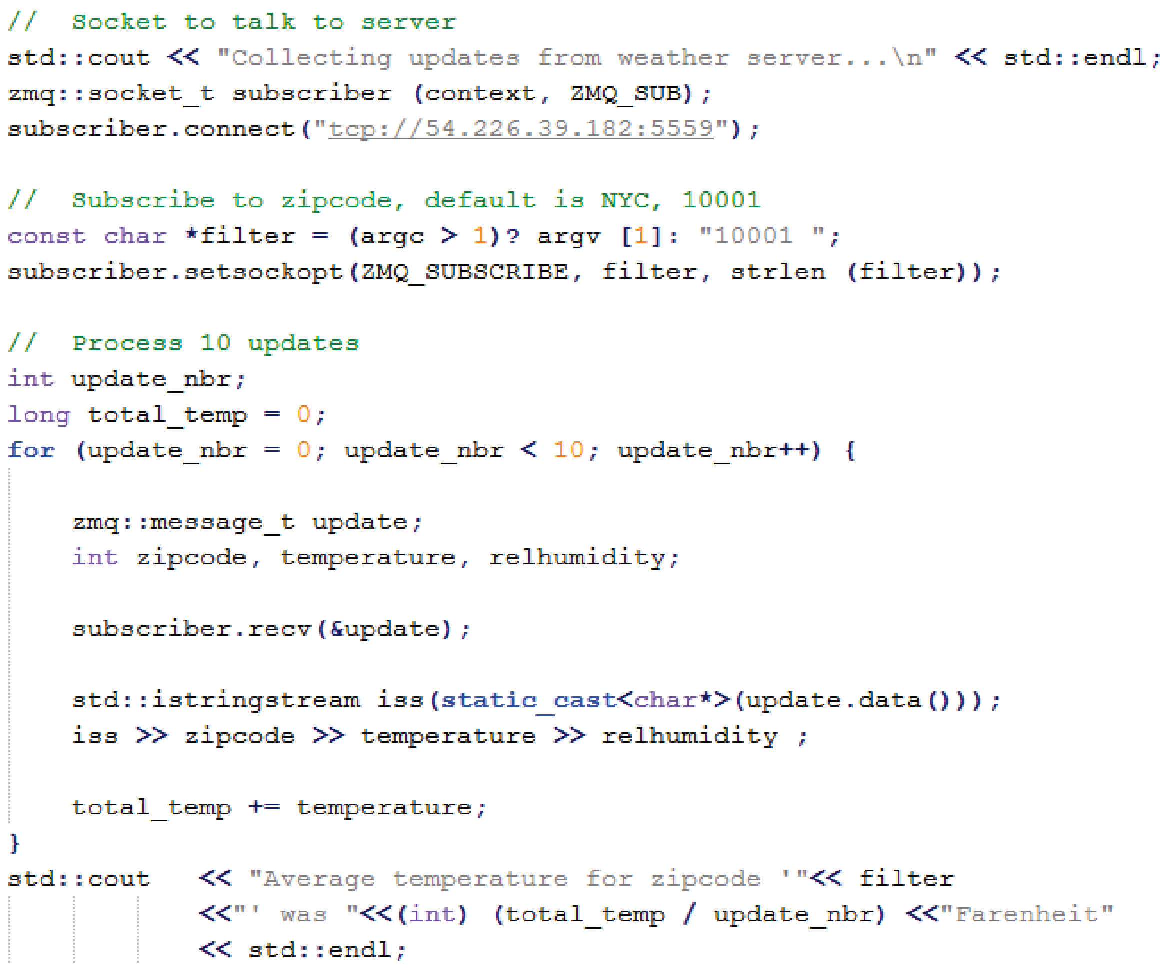
```
//  Prepare our context and publisher
zmq::context_t context (1);
zmq::socket_t publisher (context, ZMQ_PUB);
publisher.connect("tcp://54.226.39.182:5556");
publisher.bind("ipc://weather.ipc");

//  Initialize random number generator
srandom ((unsigned) time (NULL));
while (1) {

    int zipcode, temperature, relhumidity;

    //  Get values that will fool the boss
    zipcode     = within (100000);
    temperature = within (215) - 80;
    relhumidity = within (50) + 10;

    //  Send message to all subscribers
    zmq::message_t message(20);
    snprintf ((char *) message.data(), 20 ,
        "%05d %d %d", zipcode, temperature, relhumidity);
    publisher.send(message);
}
```

Figure 5.1: Weather Update Publisher

Rather than processing temperature and relative humidity updates for zipcode 10001, the program only performs calculations on the temperature updates. After receiving 10 updates for zipcode 10001, the program calculates the average temperature and reports it to the user on the screen.

5.4.3 Proxy Program.

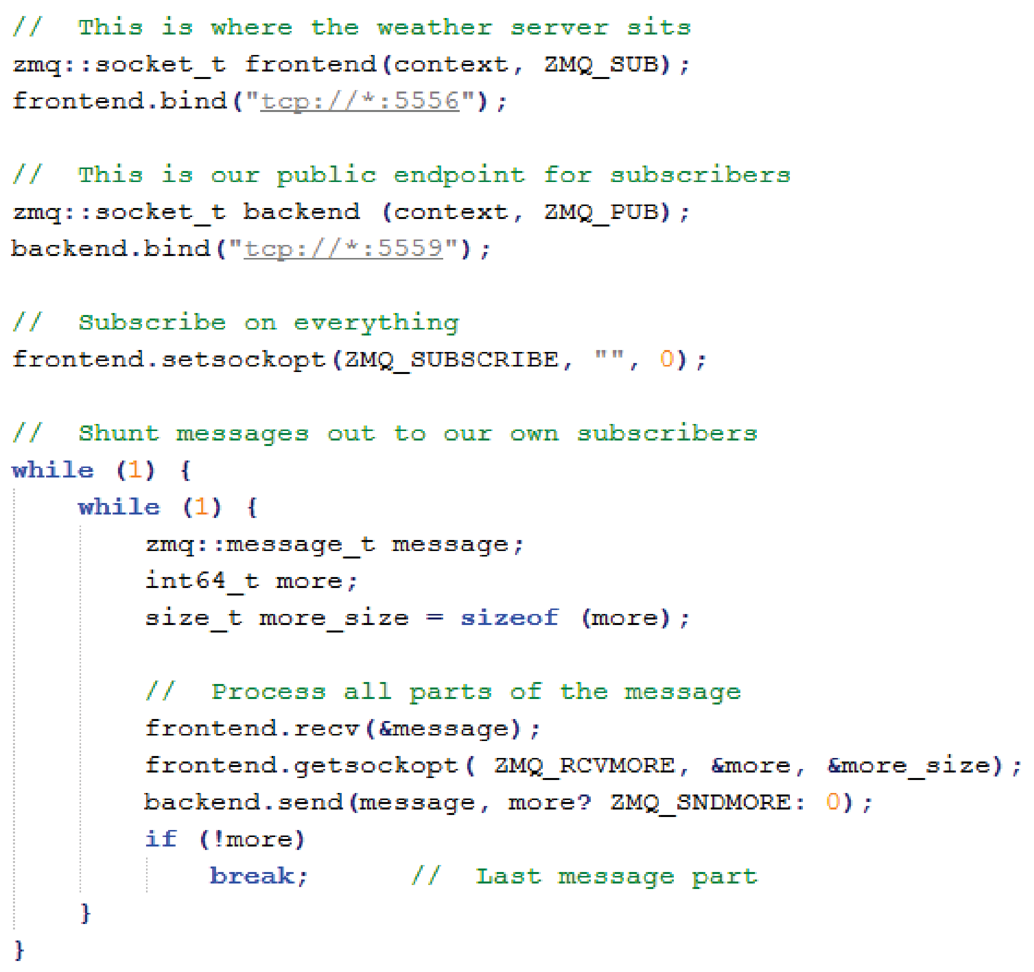
Figure 5.3 shows the applicable portion of the weather update proxy program. The program begins with the socket connection to the publisher. ZeroMQ uses a connect and bind construct. Since the publisher connects to the proxy, the proxy then binds to port



```
//  Socket to talk to server
std::cout << "Collecting updates from weather server...\n" << std::endl;
zmq::socket_t subscriber (context, ZMQ_SUB);
subscriber.connect("tcp://54.226.39.182:5559");

//  Subscribe to zipcode, default is NYC, 10001
const char *filter = (argc > 1)? argv [1]: "10001 ";
subscriber.setsockopt(ZMQ_SUBSCRIBE, filter, strlen (filter));

//  Process 10 updates
int update_nbr;
long total_temp = 0;
for (update_nbr = 0; update_nbr < 10; update_nbr++) {

    zmq::message_t update;
    int zipcode, temperature, relhumidity;

    subscriber.recv(&update);

    std::istringstream iss(static_cast<char*>(update.data()));
    iss >> zipcode >> temperature >> relhumidity ;

    total_temp += temperature;
}
std::cout   << "Average temperature for zipcode '"<< filter
            <<"' was "<<(int) (total_temp / update_nbr) <<"Farenheit"
            << std::endl;
```

Figure 5.2: Weather Update Subscriber

5556 on its local machine. This connection establishes a subscription to the data sent by the publisher. The proxy follows the same procedure to for the subscriber, binding to port 5559 in order to publish data received from the publisher out to subscribers. The proxy subscribes to everything from the publisher, and subscribers can choose to filter out unwanted data. The proxy then executes the process of receiving messages on the front end, and relaying those messages out of the back end.



```
//  This is where the weather server sits
zmq::socket_t frontend(context, ZMQ_SUB);
frontend.bind("tcp://*:5556");

//  This is our public endpoint for subscribers
zmq::socket_t backend (context, ZMQ_PUB);
backend.bind("tcp://*:5559");

//  Subscribe on everything
frontend.setsockopt(ZMQ_SUBSCRIBE, "", 0);

//  Shunt messages out to our own subscribers
while (1) {
    while (1) {
        zmq::message_t message;
        int64_t more;
        size_t more_size = sizeof (more);

        //  Process all parts of the message
        frontend.recv(&message);
        frontend.getsockopt( ZMQ_RCVMORE, &more, &more_size);
        backend.send(message, more? ZMQ_SNDMORE: 0);
        if (!more)
            break;      //  Last message part
    }
}
```

Figure 5.3: Weather Update Proxy

5.5 Case Study Results

After running the three programs on the machines in the cloud, the subscriber was able to receive data from the publisher via the proxy machine. Creating two similar publisher and corresponding subscriber programs expanded the architecture to three publishers and three subscribers that all connect to one proxy machine. All subscribes were still able to receive updates from publishers. Additionally, the Boeing host machine was able to run the subscriber program and receive updates from the cloud. This proof of concept experiment shows that real-time distributed applications can be modified if necessary to operate within the restrictions of the cloud environment. Incorporating ZeroMQ's Publisher-Subscriber messaging framework into the the OpenEaagles platform serves Boeing and Morgan State University's data analysis needs.

VI. Conclusion

6.1 Investigative Questions

1. Question What techniques do researchers use to conduct network experimentation?

   Answer Researchers use simulation, emulation and integrated network experimentation techniques to conduct network experimentation.

2. Question What tools are used to implement these techniques?

   Answer Simulation tools such as OMNet++, NetSim, GNS3, NS-3 and OPNET are popular choices to run simulations. Tools such as Emulab, Planetlab and VINI are popular emulation platform choices. The public cloud can also be leveraged to combine simulation and emulation techniques.

3. Question What effect on cloud testbed performance does physical distance between virtual machine (VM) instances have?

   Answer Physical distance between virtual machines increases network latency. The baseline platform has the lowest amount of latency because both virtual machines reside on the same host. Cloud Configurations 1 and 2 have slightly higher latency because they consists of virtual machines that are in the same availability zone, but not necessarily on the same host. Cloud Configurations 3 and 4 have the highest latency values because they consist of virtual machines that reside in different availability zones. Ultimately, all cloud platforms exhibit latency values that are well within production network expectations.

4. Question What is the difference in round-trip time, packet loss, delay, jitter and throughput among various Amazon cloud platforms, compared to the baseline platform?

Answer Round-trip time for all cloud platforms are higher than the baseline network, although they are well within production network expectations. Packet loss and delay values are all comparable to baseline platform performance. Jitter values for cloud platforms are either comparable to or slightly below baseline performance values, but all well within production network expectations. Throughput values for all cloud platforms exceed baseline performance.

5. Question Can the public cloud serve as a testbed to perform integrated network experimentation?

   Answer Yes. Cloud testbed performance metrics show that network performance in a cloud environment mimics the performance of networks constructed outside of the cloud. Therefore, the cloud is a suitable for conducting integrated network experimentation.

6.2 Future Research

The next step toward building a DoD testbed in the cloud is to incorporate Wide Area Network (WAN) emulation techniques to simulate various types of network links such as the low bandwidth, high latency satellite links found on Department of Defense (DoD) tactical networks. Currently, the cloud allows users to create virtual machine instances in the cloud, but the link capacities between those machines equate to whatever the cloud infrastructure can deliver. There is no native capability in the cloud to emulate links that make performance worse than what the cloud is capable of delivering. Implementing this capability will allow users to create topologies that mimic the exact topology of production networks, adding more realism to experiments.

6.3 Final Thoughts

Experimental metrics and statistical analysis show that cloud performance mimics expected performance of any network. Additionally, ZeroMQ provides a framework to overcome cloud policy restrictions, and allow distributed applications to send group messages without using broadcast or multicast packets. Leveraging the resources of the public cloud provides the necessary realism without the need to purchase or maintain a separate network or emulation testbed. Therefore, the public cloud infrastructure can serve as testbed for performing integrated network experimentation.

Appendix A:
Pilot Experiments

This appendix describes the process of using a Plackett-Burman design to narrow down the 16 parameters in the study to a list of five experimental factors. The discussion begins with an introduction to parameters and experimental factors, followed by a description of the theory behind the Plackett-Burman design. After listing the required materials and equipment, the discussion proceeds with an outline of the procedures and process for completing the necessary experiments. Finally, the discussion concludes with a list of the resulting experimental factors.

A.1 Introduction

System parameters affect the performance of the System Under Test(SUT). Section 3.4 describes four workload parameters, and Section 3.6 describes 12 system parameters for a total of 16 experimental parameters. The parameters with the greatest effect on performance are chosen to be experimental factors. Each factor is assigned factor levels. For example, the levels for the traffic protocol factor are TCP, UDP and ICMP. As these factor levels change, they may produce measurable effects on performance metrics such as throughput and packet loss. This research uses a complete factorial design in order to test every possible combination at all factor levels. This research also assumes that a sufficiently small variance is observed with five repetitions of each experiment. If all 16 parameters in this study were chosen as experimental factors, even with only two levels per factor, and five repetitions, the study would require 2x2x2x2x2x2x2x2x2x2x2x2x2x2x2x2x5=327,680 experiments. This is clearly an unreasonable amount of experiments. In order to limit the number of experiments required, parameters must be narrowed down to a reasonable number of experimental factors. This study uses the Plackett-Burman design to assist with making those choices.

A.2 Theory

Plackett-Burman designs are experimental designs presented in 1946 by Robin L. Plackett and J. P. Burman for investigating the dependence of some measured quantity on a number of experimental factors [23]. Although complete factorial designs accomplish this task, the number of required experiments in a complete factorial design increases exponentially as the number of factors increase. Therefore, the idea was to find smaller designs that identify the main effects of each factor using a limited number of experiments. This study begins by considering each parameter that can be independently changed by the user as a potential factor, totaling seven potential factors. The study then uses a Plackett-Burman design where each potential factor has two levels, a +1 level and a -1 level. For example, traffic protocol has TCP as the +1 level, and UDP as the -1 level during the Plackett-Burman tests. The Plackett-Burman design for the case of two levels per factor uses the method found in 1933 by Raymond Paley for generating orthogonal matrices whose elements are either 1 or -1. These matrices of size N, where N is a multiple of four but not a power of two, shows the pattern in which to vary the factor levels in the experiment [30]. In this case, the experiment has seven parameters that can be independently varied, requiring a matrix of size 12. The matrix is of size 12 because 12 is the smallest multiple of four that is greater than seven, but not also a power of two. The number eight is a multiple of four, but since it is a power of two, it is ineligible under the Plackett-Burman design. The Plackett-Burman design assumes that interactions between the factors are negligible. When interactions between factors are not negligible, they are *confounded* with the main effects, preventing one from distinguishing between certain interactions and certain main effects [31]. Since this study uses the Plackett-Burman design as a screening mechanism only, confounding is not considered to be a problem. The Plackett-Burman design identifies the most influential factors in any way, allowing for further investigation of identified factors during the complete factorial design.

A.3 Materials and Equipment

In order to conduct the Plackett-Burman screening experiments, the following materials and equipment are needed:

1. Amazon Cloud VM Instances - Instances must be configured in the cloud according to the factor levels specified by the Plackett-Burman design. For example, the Instance Type factor has a +1 level of M1.Large and a -1 level of T1.Micro. Other parameters are set during the creation of instances to include the Amazon Region and Availability Zone. Each of the VMs run Ubuntu Server 13.04 with a 64 bit 3 GHz processor. This experiment uses the T1.Micro and M1.Large instances. The T1.Micro instances have 0.615 GB of memory, Amazon Elastic Block Store (EBS) only external storage, one virtual CPU and a very low resource reservation on the Amazon cloud network. The M1.Large instances have 15 GB of memory, 420 GB of native storage, four virtual CPUs and a high resource reservation on the Amazon cloud network. The reasoning behind the drastic differences in these two factor levels is to clearly see if the instance type has an effect on performance. Other factor levels are constructed in a similar fashion.

2. Distributed Internet Traffic Generator (D-ITG) Software - Section 2.3.2 describes D-ITG software. D-ITG software must be installed on each VM instance to generate and receive traffic workload for the experiments.

3. List of Parameters - Section 3.4 provides a list of workload parameters, and Section 3.6 provides a list of experimental parameters along with their descriptions. The goal of the Plackett-Burman design is to narrow this list to the most influential experimental factors, which will assume various factor levels during the complete

factorial experiments. Parameters that are not directly alterable by the user are not considered as candidates to become experimental factors. Therefore, this studying considers the following candidates:

(a) Availability Zone - When creating an instance in the Amazon Cloud, users are given the option to choose an Availability Zone for that instance.

(b) Packet Size Distribution - The D-ITG software allows users to set the packet size distribution.

(c) Packet Arrival Distribution - The D-ITG software allows users to set the packet arrival distribution.

(d) Traffic Protocol - The D-ITG software allows users to set the traffic protocol.

(e) Amazon Region - When creating an instance in the Amazon Cloud, users are given the option to choose an Amazon Regional location for that instance.

(f) Time of Day - This is the time of day that the experiment is conducted in Eastern Standard Time.

(g) Instance Type - When creating an instance in the Amazon Cloud, users are given the option to choose an instance type.

4. Data Input Spreadsheet Software - Software such as Microsoft Excel is used to capture the results in spreadsheet form and perform mathematical operations such as calculating averages.

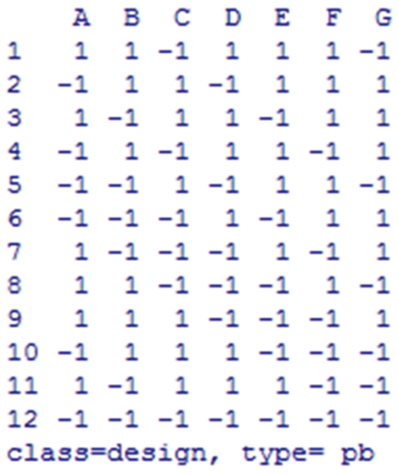
A.4 Procedures and Process

After specifying a matrix of size 12 and seven candidates for the Plackett-Burman design, the R statistical program generates the matrix shown in Figure A.1. This matrix is used to create a spreadsheet that assigns each of the seven candidates to a letter from A to G, before filling in their factor levels as shown in Figure A.2. In order to setup each experiment, each of the 12 configuration numbers represents the 12 experiments required under the Plackett-Burman design. For example, the first experimental run consists of T1. Micro VMs that are located in the Tokyo region, but in different Availability Zones, with traffic that has a packet size of 4096 at a constant rate of 1,000 packets per second, TCP protocol, and the experiment needs to be run between 11am and 2pm Eastern time. Following this process of setting up each configuration is all that is necessary to distinguish the effects of each candidate on performance metrics. Each experiment runs for 10 seconds. After designating one VM as the D-ITG sender and the other the D-ITG receiver and and configuring the traffic parameters, the results of each experimental run will be saved to the log files specified. Figure B.3 shows an example of the commands used in D-ITG. For example, the sender command specifies a destination IP address of 10.0.0.2, and tells the local sender machine to create a log file named sender.log to capture the results. Additionally, it tells the remote receiver to create a log file named receiver.log to capture the results received on the far end. The results captured in these two files should be nearly identical. The command also specifies a round-trip time meter versus one way, a duration of 10,000 milliseconds, which is equal to 10 seconds, TCP protocol, constant packet size of 4096 and a constant packet distribution of 1,000 packets per second. Using these commands in combination with the choices made while creating the VMs in the Amazon cloud allows for the construction of each of the 12 unique configurations for the Placket-Burman design.

A.5 Results

Figure A.4 shows the values of the metrics that were captured after each of the 12 configuration runs. The metrics are Throughput, Packet Loss, Delay and Jitter. After capturing these values, the next step is to return to the factor matrix. Figure A.5 shows the candidates and their factor levels of +1 and -1 depending on the configuration. The

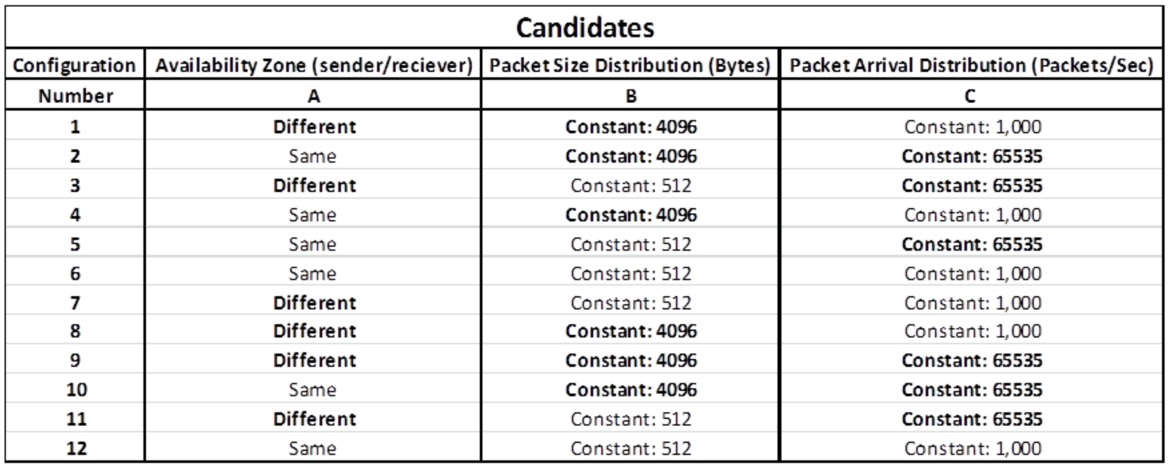
```
pb(12, 7, randomize = FALSE)
# Plackett-Burman Design for 7 factors
```



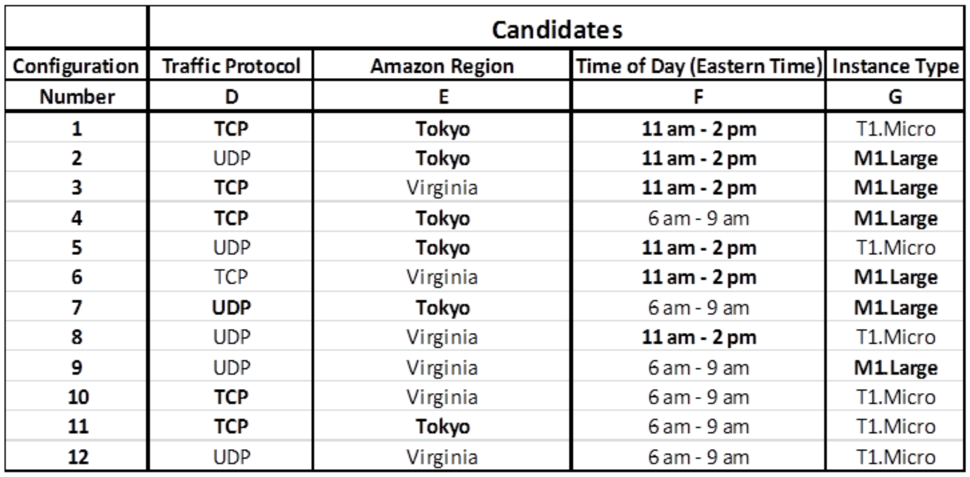
```
    A  B  C  D  E  F  G
1   1  1 -1  1  1  1 -1
2  -1  1  1 -1  1  1  1
3   1 -1  1  1 -1  1  1
4  -1  1 -1  1  1 -1  1
5  -1 -1  1 -1  1  1 -1
6  -1 -1 -1  1 -1  1  1
7   1 -1 -1 -1  1 -1  1
8   1  1 -1 -1 -1  1 -1
9   1  1  1 -1 -1 -1  1
10 -1  1  1  1 -1 -1 -1
11  1 -1  1  1  1 -1 -1
12 -1 -1 -1 -1 -1 -1 -1
class=design, type= pb
```

Figure A.1: Plackett-Burman Matrix for 7 Factors

candidate labeled *I* is the mean value, which is why it has all +1 values in the matrix. To calculate the sum of the throughput attributed to candidate A, multiply the factor level (configuration one's factor level for candidate A is +1) by the measured Throughput (configuration one's Throughput is 20.321874307). Perform this calculation for each of the configurations down candidate A's column, then sum the values to get a sum Throughput of -257.379727. Dividing this value by 12, the number of experimental runs, gives the effect of -21.4483106. For the Plackett-Burman design, the sign of the number is meaningless. Only the magnitude is considered for the effect, and larger magnitudes equal greater effects. Repeating this process for each of the remaining candidates yields the results shown in Figure A.5. The color code in Figure A.5 highlights the candidate with the largest effect for a particular metric in green. The second largest effect per metric is blue, followed by brown and peach. Since candidate C, Packet Arrival Distribution had the largest effect on both Throughput and Delay, it was the first candidate chosen as an experimental factor. The remaining factors were chosen because their effects exceeded the ones not chosen.

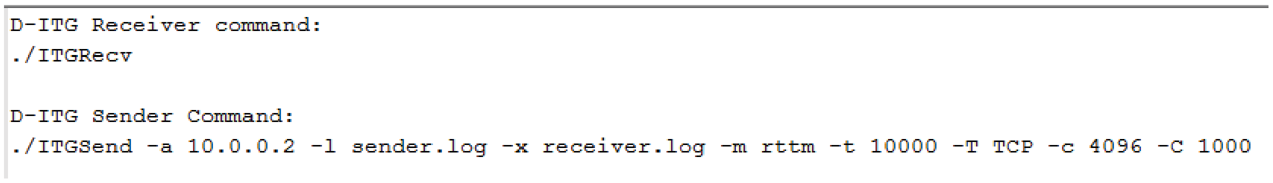


Candidates			
Configuration	Availability Zone (sender/reciever)	Packet Size Distribution (Bytes)	Packet Arrival Distribution (Packets/Sec)
Number	A	B	C
1	**Different**	**Constant: 4096**	Constant: 1,000
2	Same	**Constant: 4096**	**Constant: 65535**
3	**Different**	Constant: 512	**Constant: 65535**
4	Same	**Constant: 4096**	Constant: 1,000
5	Same	Constant: 512	**Constant: 65535**
6	Same	Constant: 512	Constant: 1,000
7	**Different**	Constant: 512	Constant: 1,000
8	**Different**	**Constant: 4096**	Constant: 1,000
9	**Different**	**Constant: 4096**	**Constant: 65535**
10	Same	**Constant: 4096**	**Constant: 65535**
11	**Different**	Constant: 512	**Constant: 65535**
12	Same	Constant: 512	Constant: 1,000



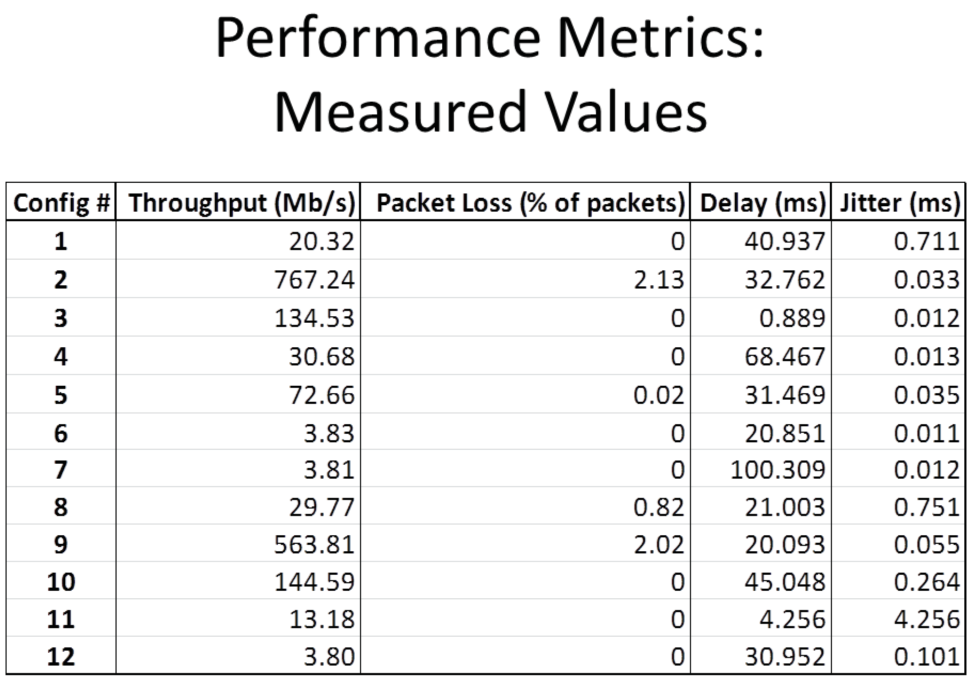
	Candidates			
Configuration	Traffic Protocol	Amazon Region	Time of Day (Eastern Time)	Instance Type
Number	D	E	F	G
1	**TCP**	**Tokyo**	**11 am - 2 pm**	T1.Micro
2	UDP	**Tokyo**	**11 am - 2 pm**	**M1.Large**
3	**TCP**	Virginia	**11 am - 2 pm**	**M1.Large**
4	**TCP**	**Tokyo**	6 am - 9 am	**M1.Large**
5	UDP	**Tokyo**	**11 am - 2 pm**	T1.Micro
6	TCP	Virginia	**11 am - 2 pm**	**M1.Large**
7	**UDP**	**Tokyo**	6 am - 9 am	**M1.Large**
8	UDP	Virginia	**11 am - 2 pm**	T1.Micro
9	UDP	Virginia	6 am - 9 am	**M1.Large**
10	**TCP**	Virginia	6 am - 9 am	T1.Micro
11	**TCP**	**Tokyo**	6 am - 9 am	T1.Micro
12	UDP	Virginia	6 am - 9 am	T1.Micro

Figure A.2: Candidates and Factor Levels



```
D-ITG Receiver command:
./ITGRecv

D-ITG Sender Command:
./ITGSend -a 10.0.0.2 -l sender.log -x receiver.log -m rttm -t 10000 -T TCP -c 4096 -C 1000
```

Figure A.3: D-ITG Commands



Performance Metrics: Measured Values

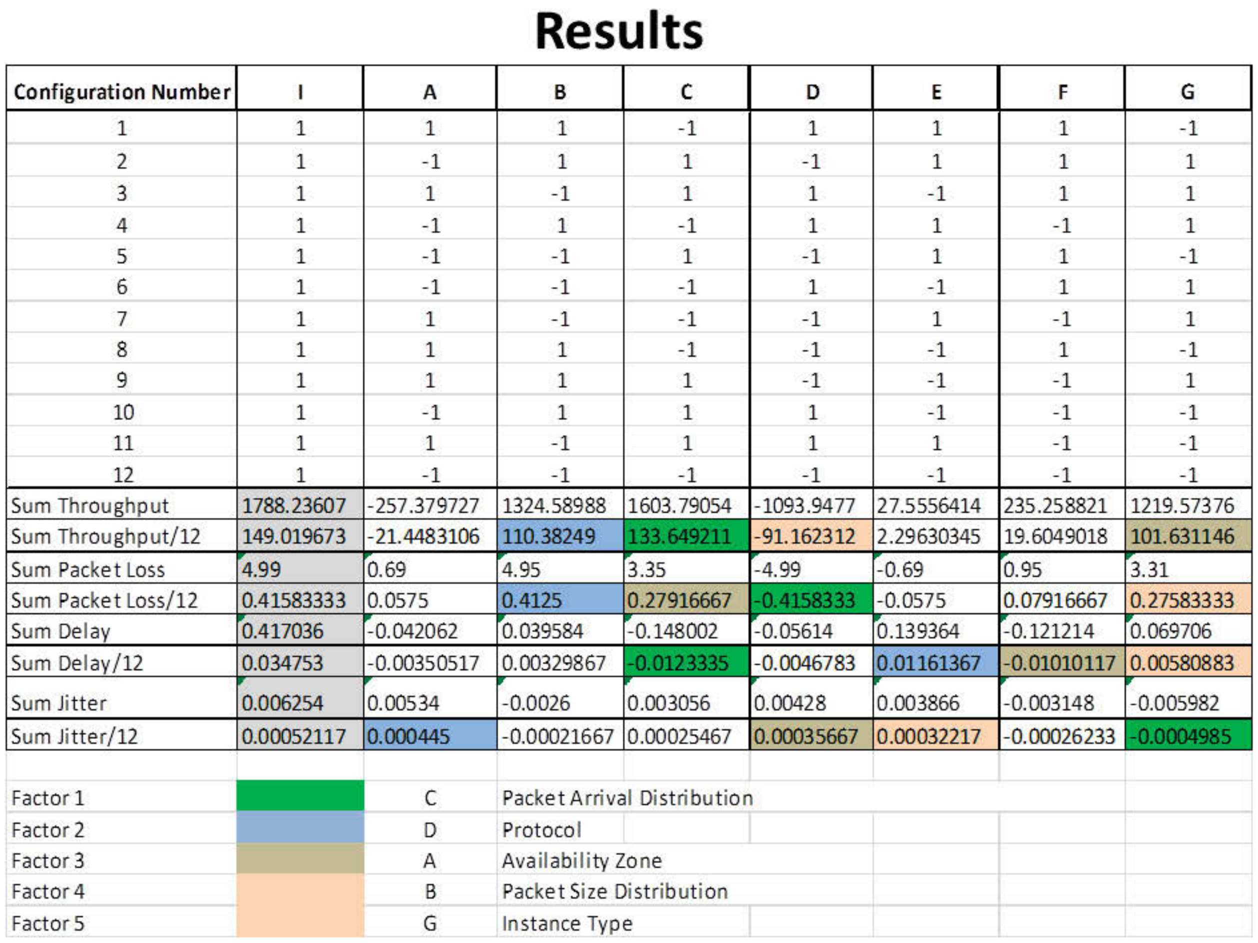
Config #	Throughput (Mb/s)	Packet Loss (% of packets)	Delay (ms)	Jitter (ms)
1	20.32	0	40.937	0.711
2	767.24	2.13	32.762	0.033
3	134.53	0	0.889	0.012
4	30.68	0	68.467	0.013
5	72.66	0.02	31.469	0.035
6	3.83	0	20.851	0.011
7	3.81	0	100.309	0.012
8	29.77	0.82	21.003	0.751
9	563.81	2.02	20.093	0.055
10	144.59	0	45.048	0.264
11	13.18	0	4.256	4.256
12	3.80	0	30.952	0.101

Figure A.4: Measured Values

A.6 Conclusion

After using the Plackett-Burman design to screen out candidates, here are the experimental factors:

1. Packet Arrival Distribution
2. Traffic Protocol
3. Availability Zone
4. Packet Size Distribution
5. VM Instance Type



Results

Configuration Number	I	A	B	C	D	E	F	G
1	1	1	1	-1	1	1	1	-1
2	1	-1	1	1	-1	1	1	1
3	1	1	-1	1	1	-1	1	1
4	1	-1	1	-1	1	1	-1	1
5	1	-1	-1	1	-1	1	1	-1
6	1	-1	-1	-1	1	-1	1	1
7	1	1	-1	-1	-1	1	-1	1
8	1	1	1	-1	-1	-1	1	-1
9	1	1	1	1	-1	-1	-1	1
10	1	-1	1	1	1	-1	-1	-1
11	1	1	-1	1	1	1	-1	-1
12	1	-1	-1	-1	-1	-1	-1	-1
Sum Throughput	1788.23607	-257.379727	1324.58988	1603.79054	-1093.9477	27.5556414	235.258821	1219.57376
Sum Throughput/12	149.019673	-21.4483106	110.38249	133.649211	-91.162312	2.29630345	19.6049018	101.631146
Sum Packet Loss	4.99	0.69	4.95	3.35	-4.99	-0.69	0.95	3.31
Sum Packet Loss/12	0.41583333	0.0575	0.4125	0.27916667	-0.4158333	-0.0575	0.07916667	0.27583333
Sum Delay	0.417036	-0.042062	0.039584	-0.148002	-0.05614	0.139364	-0.121214	0.069706
Sum Delay/12	0.034753	-0.00350517	0.00329867	-0.0123335	-0.0046783	0.01161367	-0.01010117	0.00580883
Sum Jitter	0.006254	0.00534	-0.0026	0.003056	0.00428	0.003866	-0.003148	-0.005982
Sum Jitter/12	0.00052117	0.000445	-0.00021667	0.00025467	0.00035667	0.00032217	-0.00026233	-0.0004985

Factor 1	🟩	C	Packet Arrival Distribution
Factor 2	🟦	D	Protocol
Factor 3	🟫	A	Availability Zone
Factor 4	🟧	B	Packet Size Distribution
Factor 5	🟧	G	Instance Type

Figure A.5: Results

Appendix B:
Baseline Experiments

This appendix describes the process of conducting performance testing on the baseline platform. These experiments define the standard to which the cloud performance is compared. The discussion begins with an explanation behind the selection of the baseline platform, followed by the materials and equipment needed to conduct the experiments. After describing the procedures and process of conducting the experiments, results are presented along with a concluding statement regarding the results.

B.1 Introduction and Theory

Amazon Elastic Compute Cloud (EC2) uses Xen virtualization to deliver cost-effective, enterprise-class platforms to power user applications [32]. Xen virtualization allows Amazon to separate the logical desktop, in the form of a virtual machine, from the physical machine. Since the cloud environment uses virtualization to create one or more virtual machines on a single host, a logical choice for a baseline network is a network that uses a similar construct. This study uses a baseline network created with VMWare virtual machines (VMs) running on a single host machine. Software such as VMWare and VirtualBox have long-standing reputations in the networking community for delivering virtualization in this fashion. In order to test whether virtual machine (VM) instances in the cloud behave as one would expect VMs to behave on any network, their performance is compared to the performance of VMs on the baseline network.

B.2 Materials and Equipment

In order to conduct the baseline network experiments, the following materials and equipment are needed:

1. Host Machine - The host machine has the following attributes:

    (a) Windows 7 Enterprise Service Pack 1
    (b) 8 GB Memory
    (c) 64-bit Operating System
    (d) Intel Xeon dual 3.00 GHz processors

2. VMWare Software - The host machine runs VMWare Desktop 9.0.0

3. Two Virtual Machines - The sender and receiver virtual machines created in VMWare have the following attributes:

    (a) Ubuntu 13.0.4 64-bit operating system
    (b) Intel Xeon 3 GHz processor
    (c) 1 Virtual CPU
    (d) 1 GB Memory
    (e) 410 GB Storage

4. Distributed Internet Traffic Generator(D-ITG) Software - Section 2.3.2 describes D-ITG software. D-ITG software must be installed on each VM instance to generate and receive traffic workload for the experiments.

5. Experimental Factors - Table 3.1 provides a list of experimental factors along with their corresponding levels. Two of the five factors, Instance Type and Availability Zone, only apply to the Amazon cloud environment, which is why the baseline network considers the following three experimental factors:

    (a) Packet Size Distribution - The D-ITG software allows users to set the packet size distribution.
    (b) Packet Arrival Distribution - The D-ITG software allows users to set the packet arrival distribution.

(c) Traffic Protocol - The D-ITG software allows users to set the traffic protocol.

6. Data Input Spreadsheet Software - Software such as Microsoft Excel is used to capture the results in spreadsheet form and perform mathematical operations such as calculating averages.

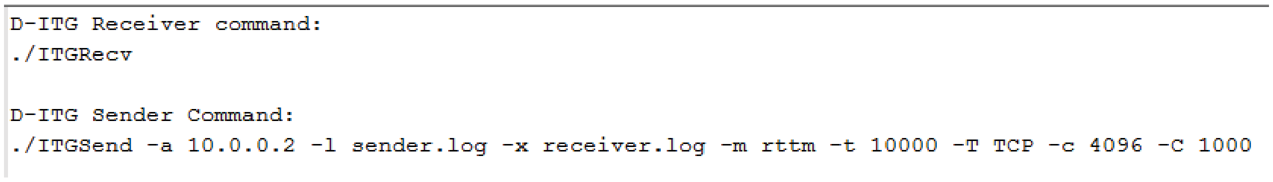
B.3 Procedures and Process

B.3.1 PING.

Network latency provides insight regarding the quality of service provided on a network. PING measures the Round-Trip Time (RTT), providing a measure of network latency. In order to measure network latency on the baseline network, the PING utility is run from one VM to the other VM, and statistics are captured regarding latency and packet loss. The test is repeated five times, for a total of six experimental runs.

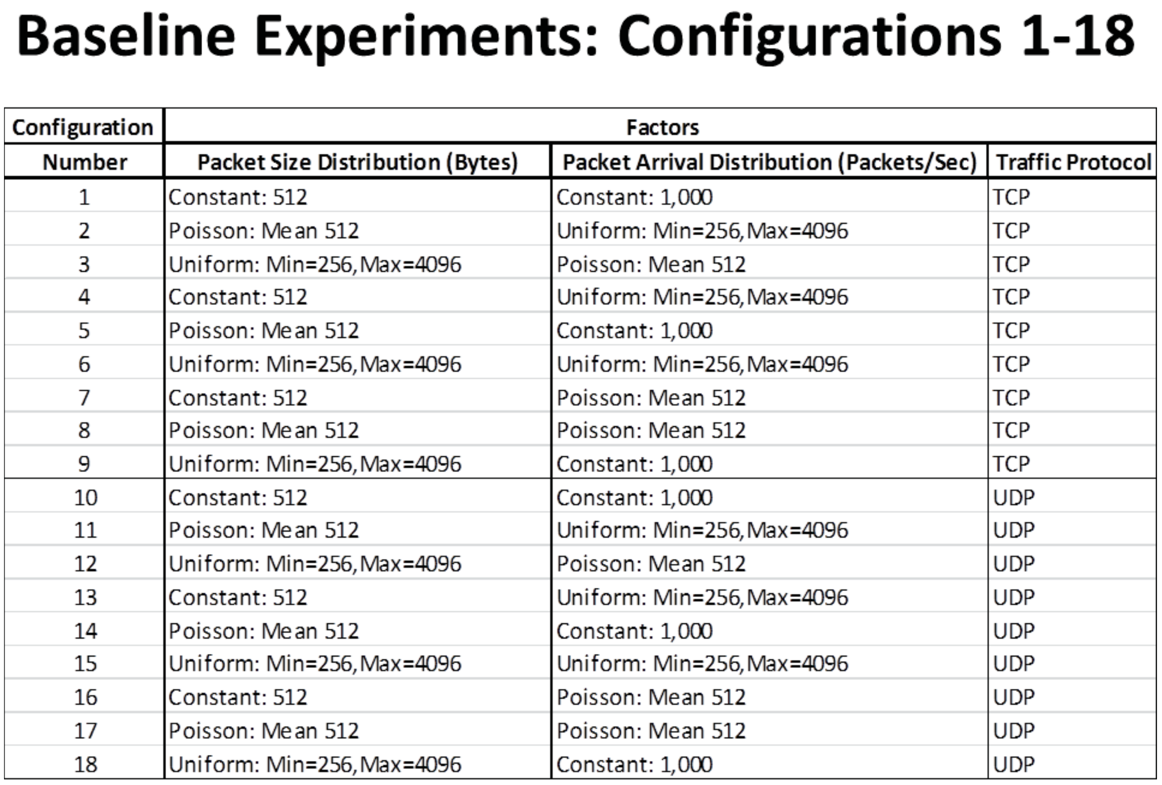
B.3.2 Workload Tests.

Capturing various performance metrics under diverse workload provides further insight regarding network performance. Workload tests are constructed using the D-ITG software. D-ITG needs to run on both sender and receiver VMs. Figure B.1 illustrates an example of the commands used in D-ITG. Experimental runs follow the configurations outlined in Figure B.2. The study uses 18 traffic workload configurations, that are standardized across all platforms to allow comparison. Traffic configurations 1-18 represent characteristics of traffic that is likely to be found on a real network. For example, as shown in Figure B.2, Configuration 1 has TCP traffic with a constant packet size of 512 bytes and a constant packet arrival distribution of 1,000 packets per second. Configuration 1 for each of the four cloud platforms has the same traffic load, allowing a standard comparison among all platforms. Experiments under each configuration are repeated five times, for a total of six experimental runs per configuration.



```
D-ITG Receiver command:
./ITGRecv

D-ITG Sender Command:
./ITGSend -a 10.0.0.2 -l sender.log -x receiver.log -m rttm -t 10000 -T TCP -c 4096 -C 1000
```

Figure B.1: D-ITG Commands



Baseline Experiments: Configurations 1-18

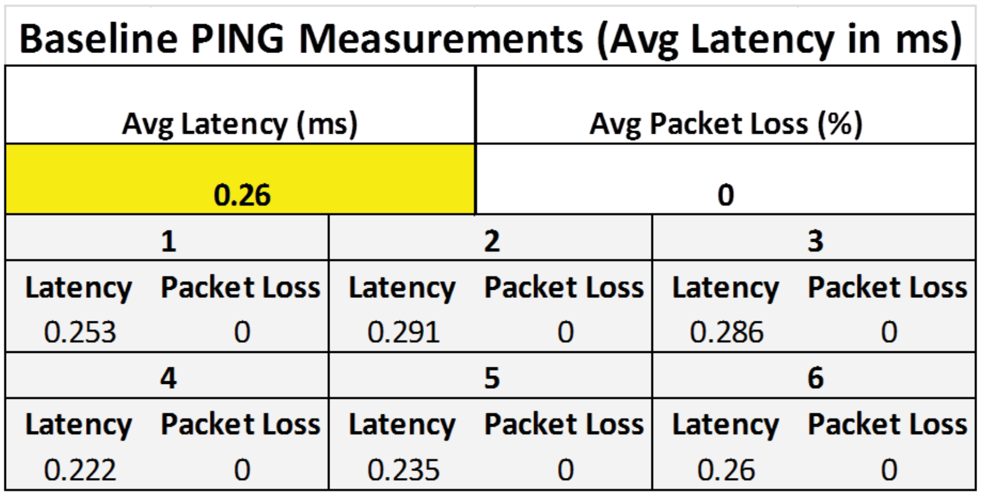
Configuration	Factors		
Number	Packet Size Distribution (Bytes)	Packet Arrival Distribution (Packets/Sec)	Traffic Protocol
1	Constant: 512	Constant: 1,000	TCP
2	Poisson: Mean 512	Uniform: Min=256, Max=4096	TCP
3	Uniform: Min=256, Max=4096	Poisson: Mean 512	TCP
4	Constant: 512	Uniform: Min=256, Max=4096	TCP
5	Poisson: Mean 512	Constant: 1,000	TCP
6	Uniform: Min=256, Max=4096	Uniform: Min=256, Max=4096	TCP
7	Constant: 512	Poisson: Mean 512	TCP
8	Poisson: Mean 512	Poisson: Mean 512	TCP
9	Uniform: Min=256, Max=4096	Constant: 1,000	TCP
10	Constant: 512	Constant: 1,000	UDP
11	Poisson: Mean 512	Uniform: Min=256, Max=4096	UDP
12	Uniform: Min=256, Max=4096	Poisson: Mean 512	UDP
13	Constant: 512	Uniform: Min=256, Max=4096	UDP
14	Poisson: Mean 512	Constant: 1,000	UDP
15	Uniform: Min=256, Max=4096	Uniform: Min=256, Max=4096	UDP
16	Constant: 512	Poisson: Mean 512	UDP
17	Poisson: Mean 512	Poisson: Mean 512	UDP
18	Uniform: Min=256, Max=4096	Constant: 1,000	UDP

Figure B.2: Baseline Experiment Configurations: 1-18

B.4 Results

B.4.1 PING Test Results.

Figure B.3 shows the results from the six experimental PING runs. The average latency over all six experiments was 0.258 ms, and the average packet loss percentage was 0 percent. Ping tests from each cloud configuration are compared to these values in Chapter IV.



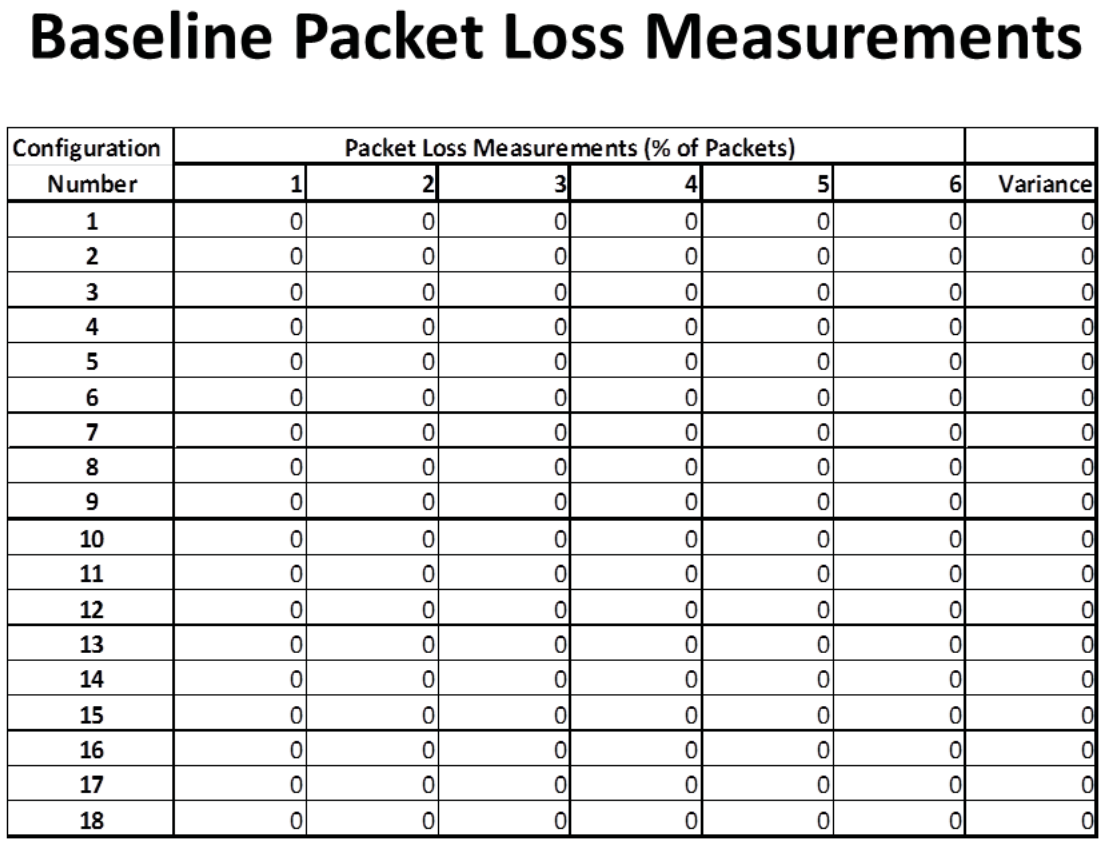
Baseline PING Measurements (Avg Latency in ms)

Avg Latency (ms)				Avg Packet Loss (%)	
0.26				**0**	
1		**2**		**3**	
Latency	Packet Loss	Latency	Packet Loss	Latency	Packet Loss
0.253	0	0.291	0	0.286	0
4		**5**		**6**	
Latency	Packet Loss	Latency	Packet Loss	Latency	Packet Loss
0.222	0	0.235	0	0.26	0

Figure B.3: Baseline Ping Results

B.4.2 Workload Test Results.

Figure B.4 shows the packet loss values for each experimental run on the baseline network, along with the variance associated with the experimental runs. Figures B.5, B.6 and B.7 show the same information for delay, jitter and throughput respectively. Figure B.8 shows the mean values for all metrics for each of the 18 experimental configurations. These mean values are compared with the mean values from each of the cloud platforms in Chapter IV.



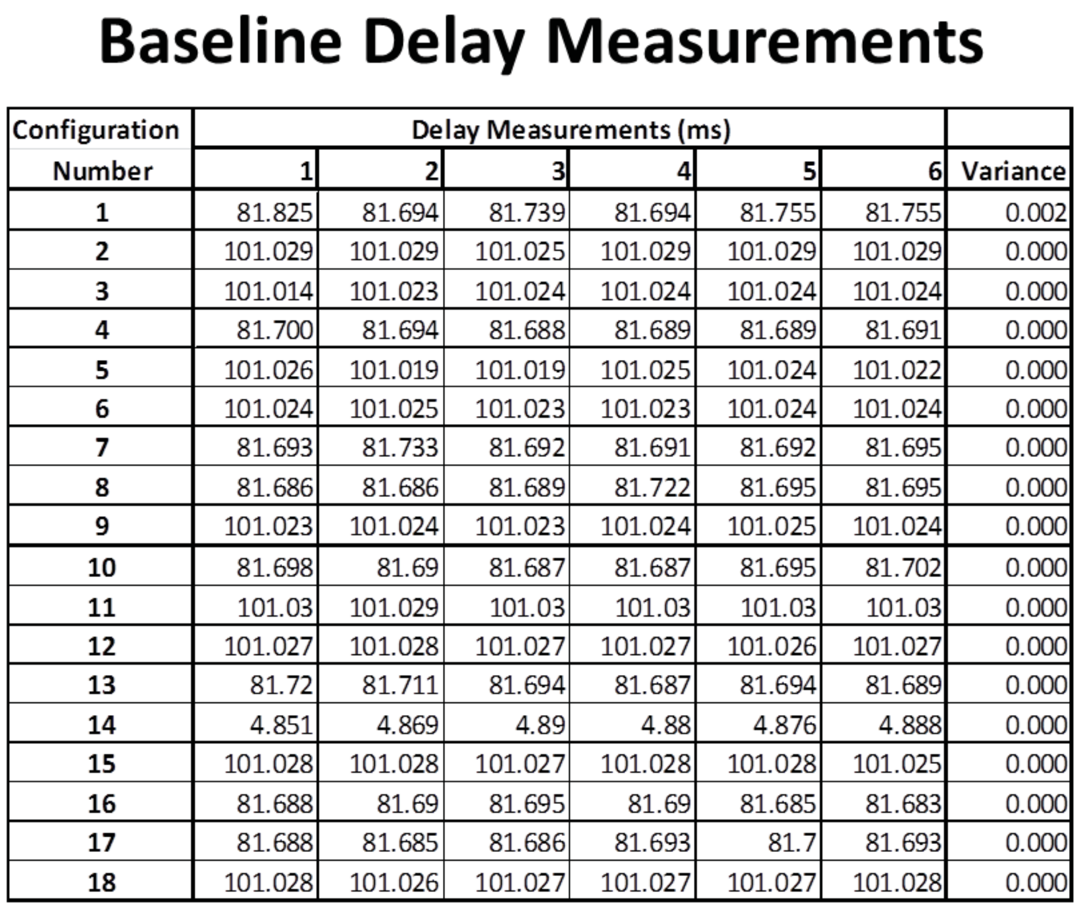
Baseline Packet Loss Measurements

Configuration Number	Packet Loss Measurements (% of Packets)						
	1	2	3	4	5	6	Variance
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0

Figure B.4: Baseline Packet Loss

B.5 Conclusion

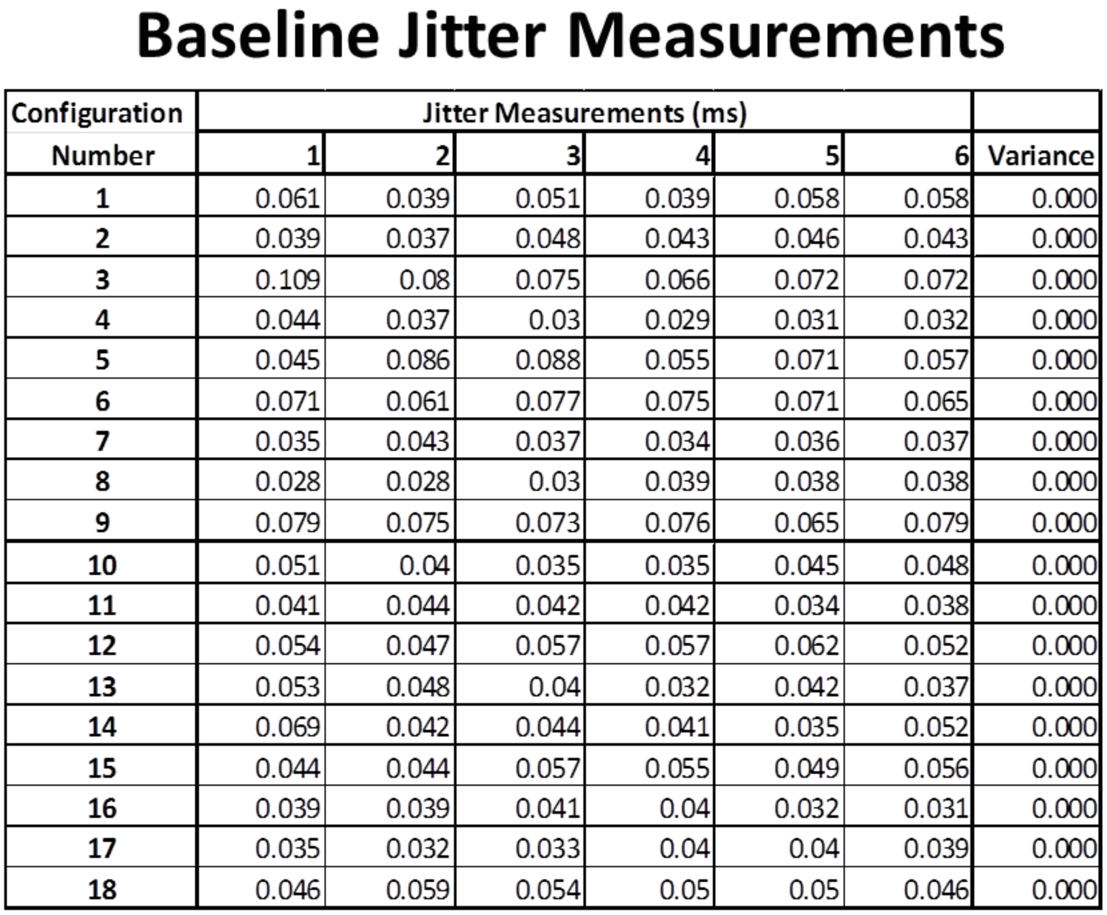
Experimental runs have sufficiently small variance to allow adequate comparison of mean values. These configurations represent typical traffic expected on a real network.



Baseline Delay Measurements

Configuration	Delay Measurements (ms)						
Number	1	2	3	4	5	6	Variance
1	81.825	81.694	81.739	81.694	81.755	81.755	0.002
2	101.029	101.029	101.025	101.029	101.029	101.029	0.000
3	101.014	101.023	101.024	101.024	101.024	101.024	0.000
4	81.700	81.694	81.688	81.689	81.689	81.691	0.000
5	101.026	101.019	101.019	101.025	101.024	101.022	0.000
6	101.024	101.025	101.023	101.023	101.024	101.024	0.000
7	81.693	81.733	81.692	81.691	81.692	81.695	0.000
8	81.686	81.686	81.689	81.722	81.695	81.695	0.000
9	101.023	101.024	101.023	101.024	101.025	101.024	0.000
10	81.698	81.69	81.687	81.687	81.695	81.702	0.000
11	101.03	101.029	101.03	101.03	101.03	101.03	0.000
12	101.027	101.028	101.027	101.027	101.026	101.027	0.000
13	81.72	81.711	81.694	81.687	81.694	81.689	0.000
14	4.851	4.869	4.89	4.88	4.876	4.888	0.000
15	101.028	101.028	101.027	101.028	101.028	101.025	0.000
16	81.688	81.69	81.695	81.69	81.685	81.683	0.000
17	81.688	81.685	81.686	81.693	81.7	81.693	0.000
18	101.028	101.026	101.027	101.027	101.027	101.028	0.000

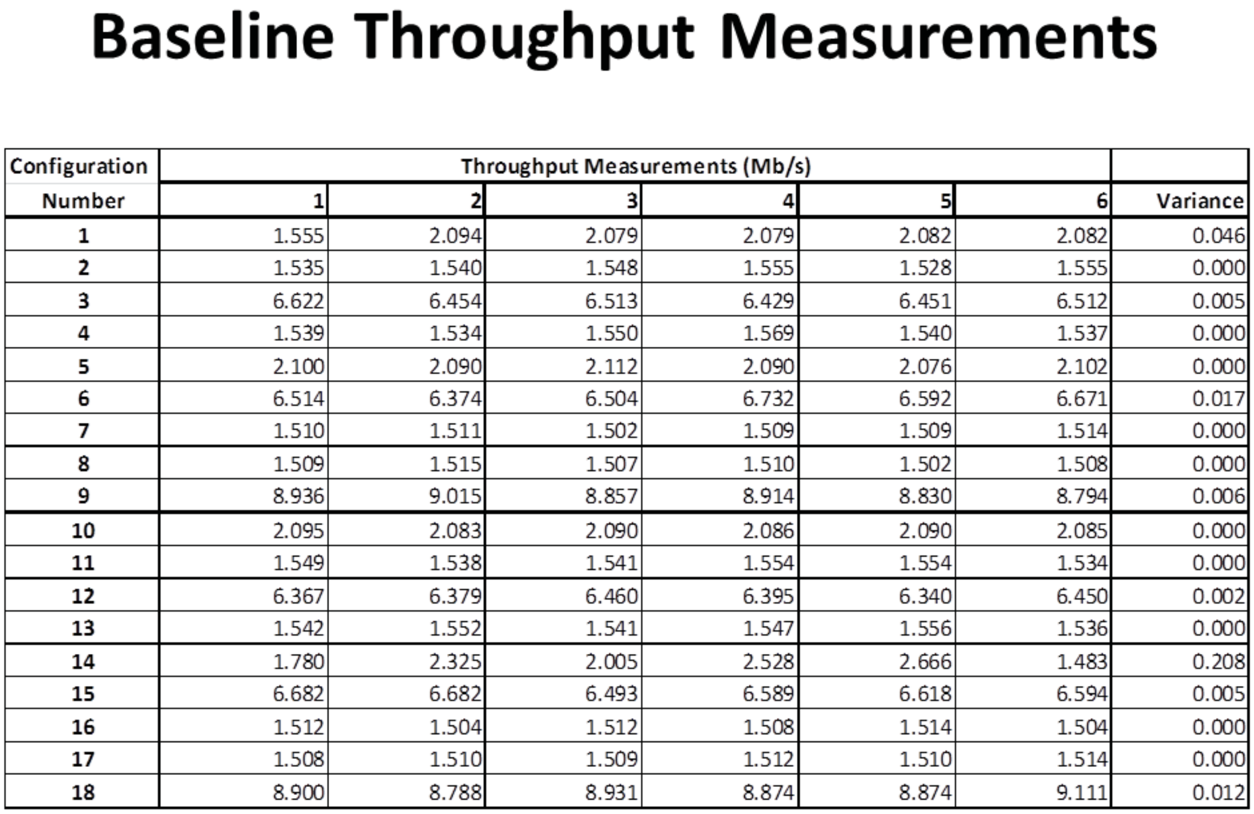
Figure B.5: Baseline Delay



Baseline Jitter Measurements

Configuration	Jitter Measurements (ms)						
Number	1	2	3	4	5	6	Variance
1	0.061	0.039	0.051	0.039	0.058	0.058	0.000
2	0.039	0.037	0.048	0.043	0.046	0.043	0.000
3	0.109	0.08	0.075	0.066	0.072	0.072	0.000
4	0.044	0.037	0.03	0.029	0.031	0.032	0.000
5	0.045	0.086	0.088	0.055	0.071	0.057	0.000
6	0.071	0.061	0.077	0.075	0.071	0.065	0.000
7	0.035	0.043	0.037	0.034	0.036	0.037	0.000
8	0.028	0.028	0.03	0.039	0.038	0.038	0.000
9	0.079	0.075	0.073	0.076	0.065	0.079	0.000
10	0.051	0.04	0.035	0.035	0.045	0.048	0.000
11	0.041	0.044	0.042	0.042	0.034	0.038	0.000
12	0.054	0.047	0.057	0.057	0.062	0.052	0.000
13	0.053	0.048	0.04	0.032	0.042	0.037	0.000
14	0.069	0.042	0.044	0.041	0.035	0.052	0.000
15	0.044	0.044	0.057	0.055	0.049	0.056	0.000
16	0.039	0.039	0.041	0.04	0.032	0.031	0.000
17	0.035	0.032	0.033	0.04	0.04	0.039	0.000
18	0.046	0.059	0.054	0.05	0.05	0.046	0.000

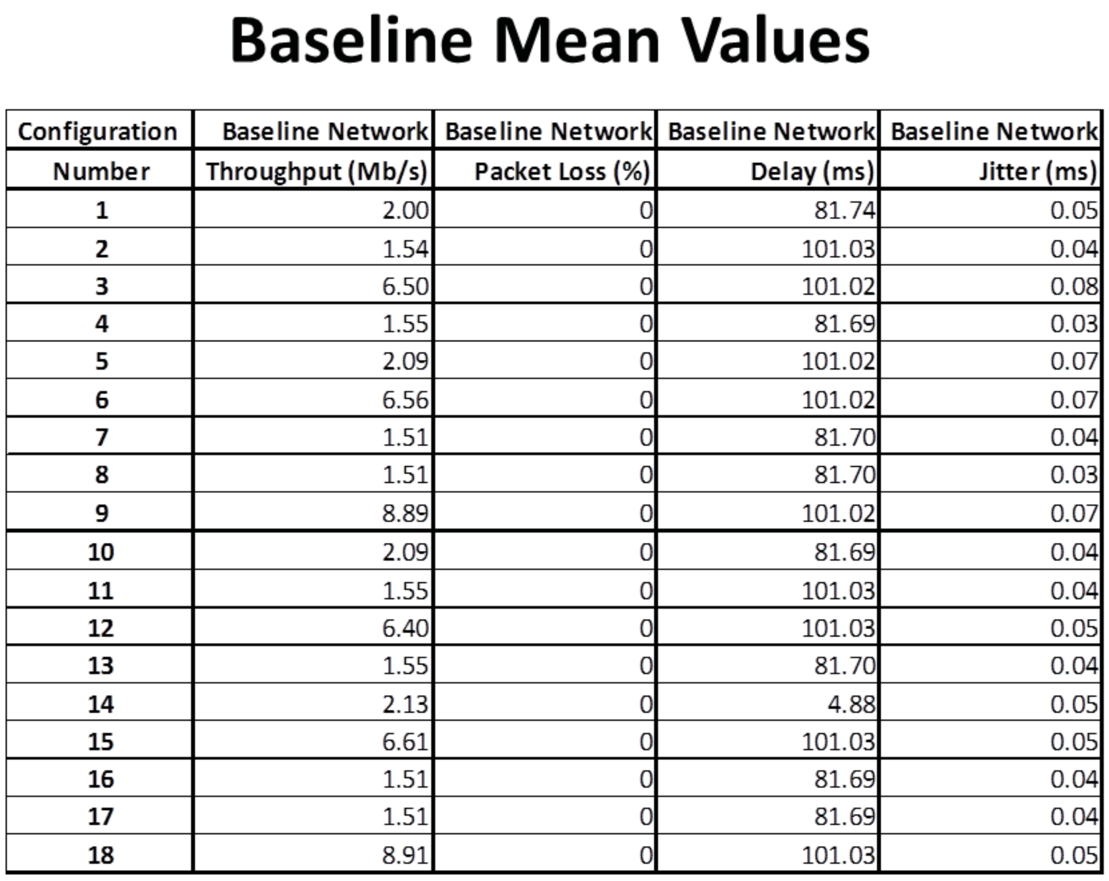
Figure B.6: Baseline Jitter



Baseline Throughput Measurements

Configuration	Throughput Measurements (Mb/s)						
Number	1	2	3	4	5	6	Variance
1	1.555	2.094	2.079	2.079	2.082	2.082	0.046
2	1.535	1.540	1.548	1.555	1.528	1.555	0.000
3	6.622	6.454	6.513	6.429	6.451	6.512	0.005
4	1.539	1.534	1.550	1.569	1.540	1.537	0.000
5	2.100	2.090	2.112	2.090	2.076	2.102	0.000
6	6.514	6.374	6.504	6.732	6.592	6.671	0.017
7	1.510	1.511	1.502	1.509	1.509	1.514	0.000
8	1.509	1.515	1.507	1.510	1.502	1.508	0.000
9	8.936	9.015	8.857	8.914	8.830	8.794	0.006
10	2.095	2.083	2.090	2.086	2.090	2.085	0.000
11	1.549	1.538	1.541	1.554	1.554	1.534	0.000
12	6.367	6.379	6.460	6.395	6.340	6.450	0.002
13	1.542	1.552	1.541	1.547	1.556	1.536	0.000
14	1.780	2.325	2.005	2.528	2.666	1.483	0.208
15	6.682	6.682	6.493	6.589	6.618	6.594	0.005
16	1.512	1.504	1.512	1.508	1.514	1.504	0.000
17	1.508	1.510	1.509	1.512	1.510	1.514	0.000
18	8.900	8.788	8.931	8.874	8.874	9.111	0.012

Figure B.7: Baseline Throughput



Baseline Mean Values

Configuration	Baseline Network	Baseline Network	Baseline Network	Baseline Network
Number	Throughput (Mb/s)	Packet Loss (%)	Delay (ms)	Jitter (ms)
1	2.00	0	81.74	0.05
2	1.54	0	101.03	0.04
3	6.50	0	101.02	0.08
4	1.55	0	81.69	0.03
5	2.09	0	101.02	0.07
6	6.56	0	101.02	0.07
7	1.51	0	81.70	0.04
8	1.51	0	81.70	0.03
9	8.89	0	101.02	0.07
10	2.09	0	81.69	0.04
11	1.55	0	101.03	0.04
12	6.40	0	101.03	0.05
13	1.55	0	81.70	0.04
14	2.13	0	4.88	0.05
15	6.61	0	101.03	0.05
16	1.51	0	81.69	0.04
17	1.51	0	81.69	0.04
18	8.91	0	101.03	0.05

Figure B.8: Baseline Mean Values

Appendix C:
Cloud Experiments

This appendix describes the process of conducting performance testing in the cloud. The cloud is divided into four unique platforms that are independently compared to the baseline platform. The discussion begins with a brief introduction and theory, followed by the materials and equipment needed to conduct the experiments. After describing the procedures and process of conducting the experiments, results are presented along with a concluding statement regarding the results.

C.1 Introduction and Theory

In order to test the feasibility of using the Amazon cloud as a networking and distributed application prototyping testbed, cloud performance must be considered. Network architectures constructed in the cloud should show similar behavior to other networks that have been traditionally used for this purpose. Each cloud platform is provided statistically equivalent workload to the workload provided to the baseline platform, performance metrics are gathered, and results are interpreted.

C.2 Materials and Equipment

In order to conduct the cloud experiments, the following materials and equipment are needed:

1. List of Factors - Here are the experimental factors:

   (a) Availability Zone - When creating an instance in the Amazon Cloud, users are given the option to choose an Availability Zone for that instance.

   (b) Packet Size Distribution - The D-ITG software allows users to set the packet size distribution.

   (c) Packet Arrival Distribution - The D-ITG software allows users to set the packet arrival distribution.

   (d) Traffic Protocol - The D-ITG software allows users to set the traffic protocol.

   (e) Instance Type - When creating an instance in the Amazon Cloud, users are given the option to choose an instance type.

2. Amazon Cloud VM Instances - Instances must be configured in the cloud according to the factor levels specified in each experimental run.

3. Distributed Internet Traffic Generator(D-ITG) Software - Section 2.3.2 describes D-ITG software. D-ITG software must be installed on each VM instance to generate and receive traffic workload for the experiments.

4. Cloud Configurations - This study uses four unique cloud platforms for testing:

   (a) Cloud Platform 1 - The first cloud platform consists of two M1.Medium instance types that are in the same Availability Zone.

   (b) Cloud Platform 2 - The second cloud platform consists of two M1.Large instance types that are in the same Availability Zone.

   (c) Cloud Platform 3 - The third cloud platform consists of two M1.Medium instance types that are in different Availability Zones.

   (d) Cloud Platform 4 - The fourth cloud platform consists of two M1.Large instance types that are in different Availability Zones.

5. Constant Parameters - While the five experimental factors will be varied during the experiments, the remaining parameters are not directly altered:

   (a) Workload Generator Random Seed - The D-ITG software randomly alters this value before each experimental run to ensure that traffic loads of the same configuration are not exactly the same, although they are statistically equivalent. For example, the software may send TCP traffic with 512 byte packets, in a Poisson distribution, with a mean value of 512 packets per second during one experimental run. During another experimental run of that same configuration, the software will not send packets in an identical pattern as the previous run, but will ensure that it achieves a mean value of 512 packets per second. This makes is difficult for a system to optimize its performance based solely on the test itself, as the system must handle a variety of different, yet statistically equivalent conditions.

(b) Operating System - All cloud VMs run Linux Ubuntu Server 13.04.

(c) Processor - All cloud VMs run a 64 bit processor.

(d) Virtual CPUs - The number of virtual CPUs in a VM varies by instance type. M1.Medium instances have one virtual CPU, and M1.Large instances have two virtual CPUs.

(e) Amazon EC2 Compute Units (ECU) - ECUs vary by instance type. M1.Medium instances have two ECUs and M1.Large instances have four ECUs.

(f) Memory - Memory capacity varies by instance type. M1.Medium instances have 3.75 GB of memory, and M1.Large instances have 7.5 GB of memory.

(g) Storage - Storage capacity varies by instance type. M1.Medium instances have 410 GB of storage, and M1.Large instances have two drives with 420 GB of storage each.

(h) Network Resource Reservation - Amazon promises a moderate network resource reservation for both the M1.Medium and M1.Large instances.

(i) Amazon Region - Since this parameter was not chosen as a factor through the Plackett-Burman design, the region is held constant. All cloud experiments take place in the Virginia region.

(j) Time of Day - Since this parameter was not chosen as a factor through the Plackett-Burman design, the experiments are not restricted to any particular time of day.

(k) Performance of Underlying Amazon Hardware - This parameter is not under the user's control. This study assumes that Amazon's underlying hardware is sufficient to perform cloud experiments.

(l) Performance of Amazon Network Management/Virtualization Software - This parameter is not under the user's control. This study assumes that Amazon's network management and virtualization software is sufficient to conduct cloud experiments.

6. Data Input Spreadsheet Software - Software such as Microsoft Excel is used to capture the results in spreadsheet form and perform mathematical operations such as calculating averages.

C.3 Procedures and Process

C.3.1 PING Tests.

PING measures the Round-Trip Time (RTT), providing a measure network latency. Since there are four unique cloud configurations, there are four different sets of PING tests. The PING tests consist of running the PING utility from one VM instance to the other VM instance and capturing the statistics that are native to the utility. Each test is repeated five times, for a total of six experimental runs per cloud configuration, or 24 total experimental runs.

C.3.2 Workload Tests.

Configurations 1-18 represent characteristics of traffic that is likely to be found on a real network. For example, packet arrival distributions range up to 4096 packets per second.

C.4 Results

C.4.1 PING Test Results.

Figure C.1 shows the results of the ping experiments. For example, Cloud Platform 1 is in the upper left hand corner. The bottom half of the rectangle shows the results of each of the six experimental runs. Cloud Platform 1 has latency values in milliseconds of 0.64, 0.62, 0.59, 0.65, 0.58 and 0.61. In all cases, Cloud Platform 1 had zero packet loss. As a result, Cloud Platform 1 has an average latency of 0.62 ms. The other three configurations are interpreted in a similar fashion. Here is a summary of the cloud configurations PING tests:

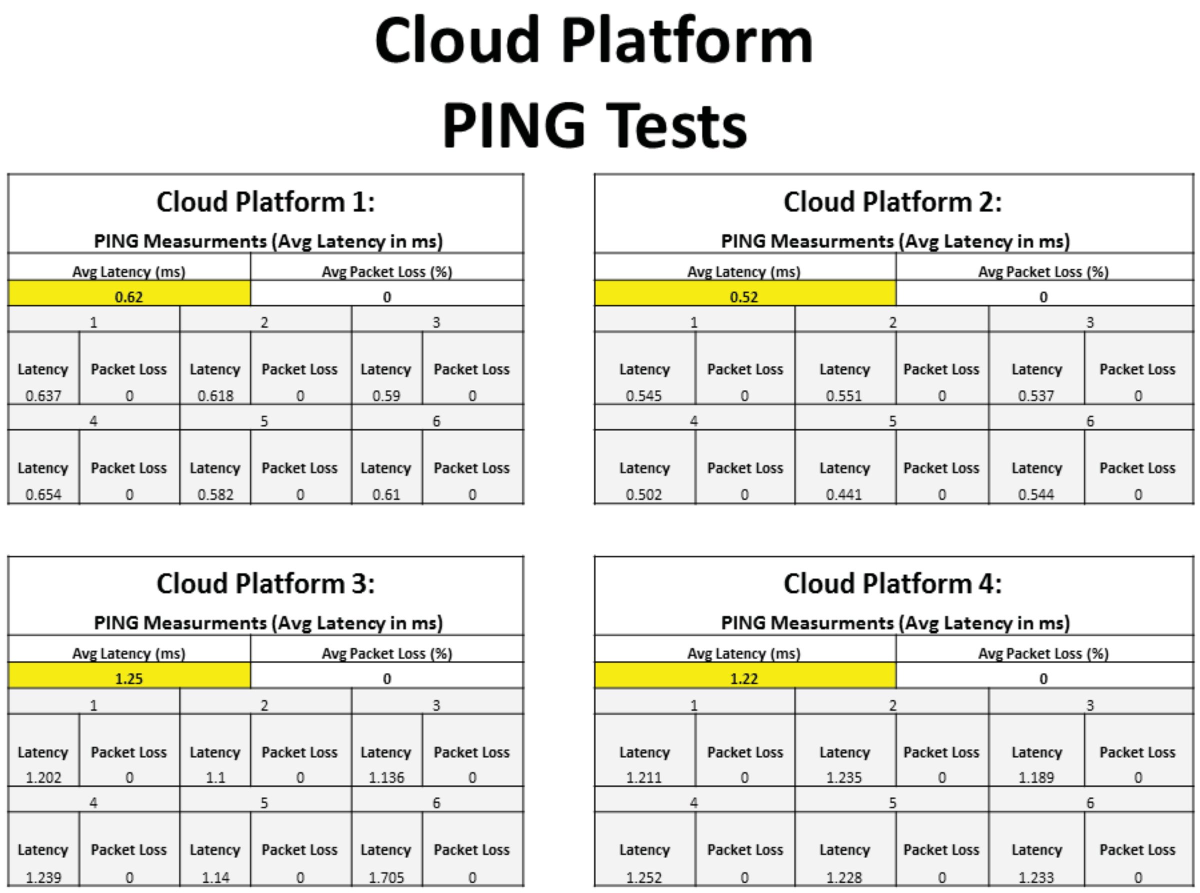
1. Cloud Platform 1 - Cloud Platform 1 has an average latency of 0.62 ms, and 0 percent packet loss.

2. Cloud Platform 2 - Cloud Platform 2 has an average latency of 0.52 ms, and 0 percent packet loss.

3. Cloud Platform 3 - Cloud Platform 3 has an average latency of 1.25 ms, and 0 percent packet loss.

4. Cloud Platform 1 - Cloud Platform 1 has an average latency of 1.22 ms, and 0 percent packet loss.

These values are compared to the PING tests from the baseline network in Chapter IV.



Cloud Platform
PING Tests

Cloud Platform 1:
PING Measurments (Avg Latency in ms)

Avg Latency (ms)	Avg Packet Loss (%)
0.62	0

1		2		3	
Latency	Packet Loss	Latency	Packet Loss	Latency	Packet Loss
0.637	0	0.618	0	0.59	0
4		5		6	
Latency	Packet Loss	Latency	Packet Loss	Latency	Packet Loss
0.654	0	0.582	0	0.61	0

Cloud Platform 2:
PING Measurments (Avg Latency in ms)

Avg Latency (ms)	Avg Packet Loss (%)
0.52	0

1		2		3	
Latency	Packet Loss	Latency	Packet Loss	Latency	Packet Loss
0.545	0	0.551	0	0.537	0
4		5		6	
Latency	Packet Loss	Latency	Packet Loss	Latency	Packet Loss
0.502	0	0.441	0	0.544	0

Cloud Platform 3:
PING Measurments (Avg Latency in ms)

Avg Latency (ms)	Avg Packet Loss (%)
1.25	0

1		2		3	
Latency	Packet Loss	Latency	Packet Loss	Latency	Packet Loss
1.202	0	1.1	0	1.136	0
4		5		6	
Latency	Packet Loss	Latency	Packet Loss	Latency	Packet Loss
1.239	0	1.14	0	1.705	0

Cloud Platform 4:
PING Measurments (Avg Latency in ms)

Avg Latency (ms)	Avg Packet Loss (%)
1.22	0

1		2		3	
Latency	Packet Loss	Latency	Packet Loss	Latency	Packet Loss
1.211	0	1.235	0	1.189	0
4		5		6	
Latency	Packet Loss	Latency	Packet Loss	Latency	Packet Loss
1.252	0	1.228	0	1.233	0

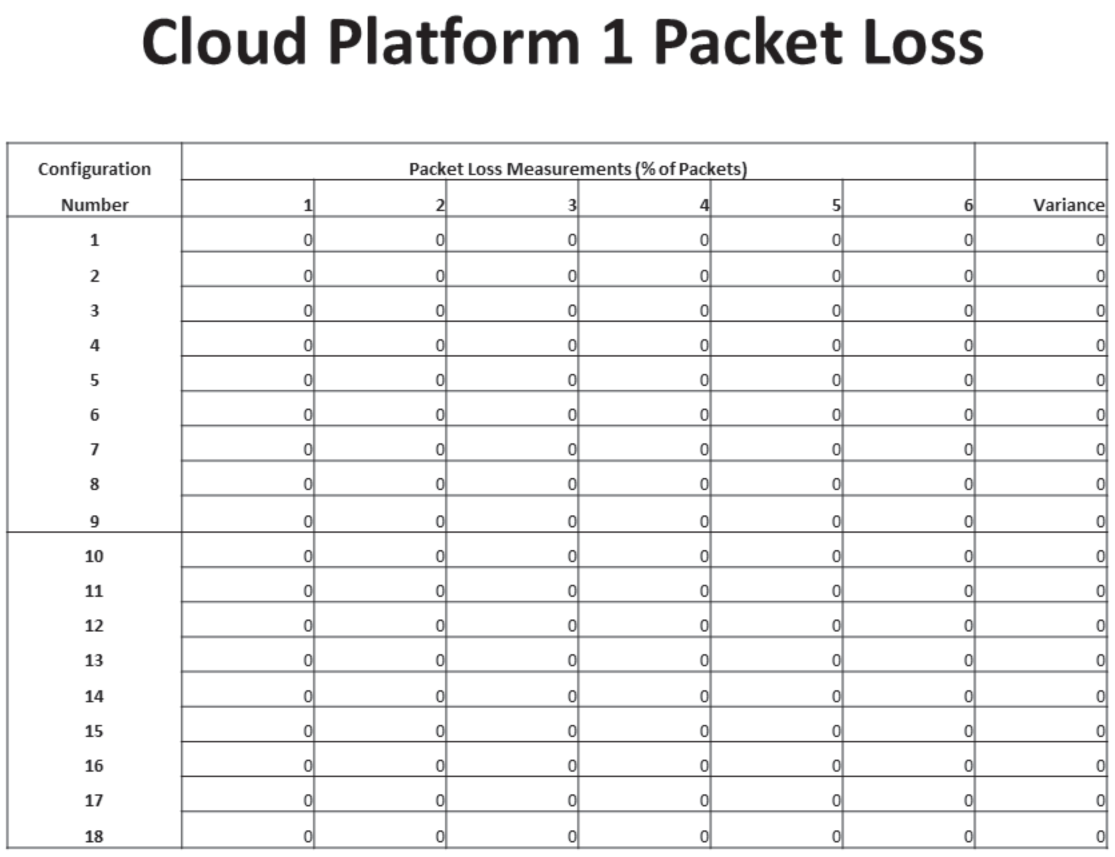
Figure C.1: Cloud Ping Tests Results

C.4.2 Workload Test Results.

This section describes the workload test results from each of the four cloud platforms. These metrics are independently compared to the metrics from the baseline platform in Chapter IV.

C.4.2.1 Cloud Configuration 1.

Figures C.2, C.3, C.4 and C.5 show the measurements for packet loss, delay, jitter and throughput respectively under Cloud Platform 1, along with corresponding variances under all 18 traffic workload configurations. This data is used to construct the mean values shown in Figure C.6. Mean values for Cloud Platform 1 are compared to the mean values of the baseline platform as well as other cloud platforms in Chapter IV.



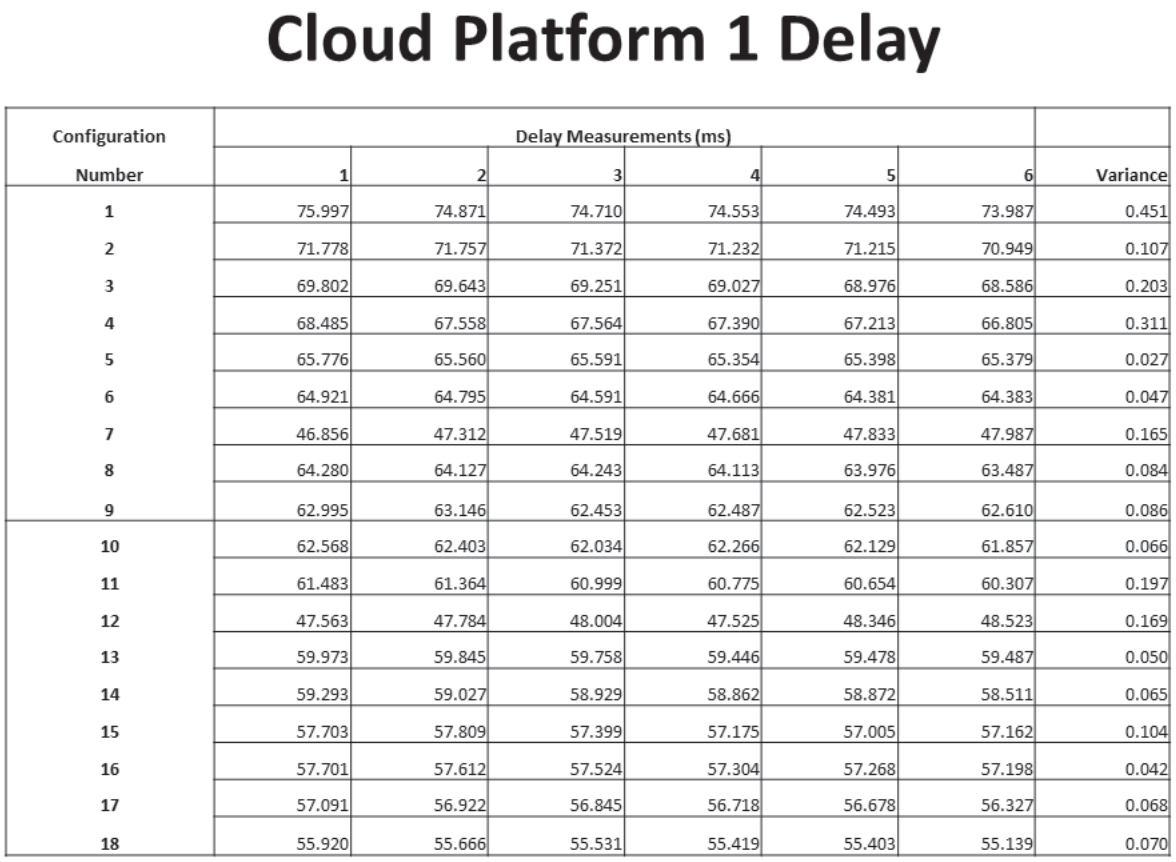
Cloud Platform 1 Packet Loss

Configuration Number	Packet Loss Measurements (% of Packets)						
	1	2	3	4	5	6	Variance
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0

Figure C.2: Cloud Platform 1 Packet Loss

C.4.2.2 Cloud Configuration 2.

Figures C.7, C.8, C.9 and C.10 show the measurements for packet loss, delay, jitter and throughput respectively under Cloud Platform 2, along with corresponding variances under all 18 traffic workload configurations. This data is used to construct the mean values



Cloud Platform 1 Delay

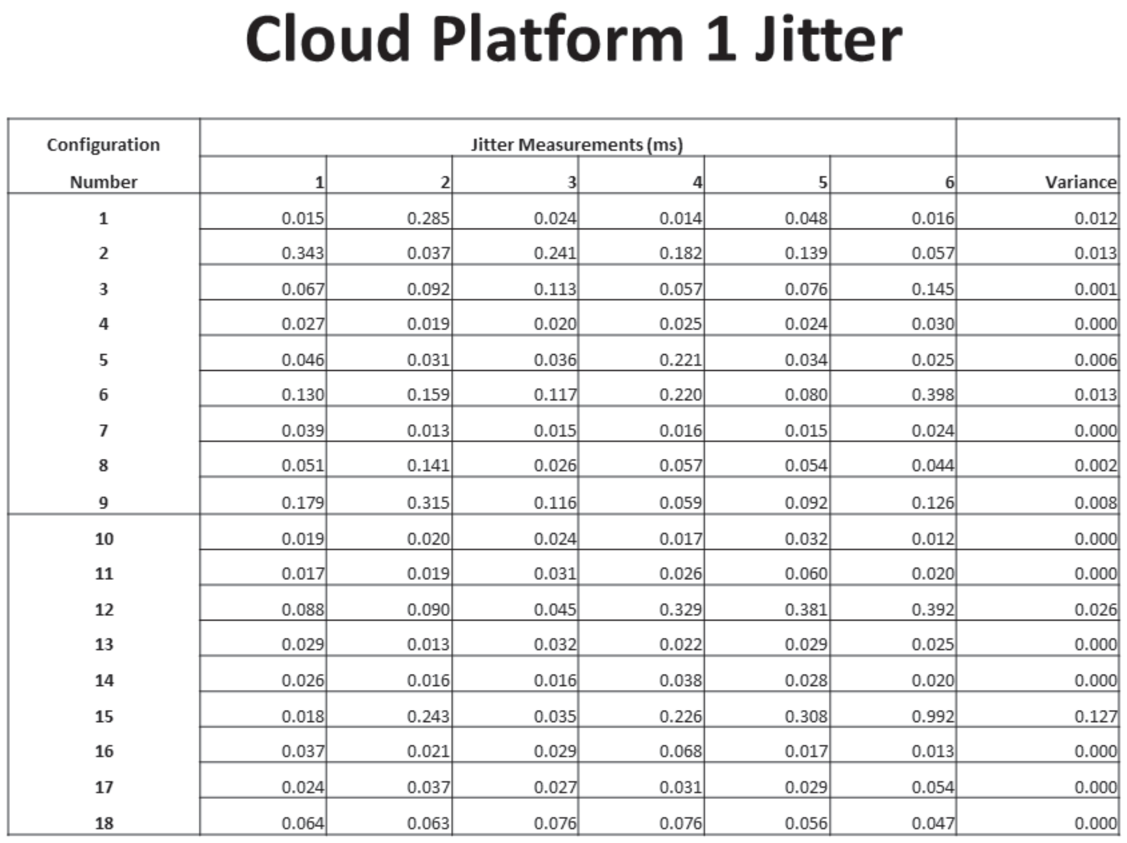
Configuration Number	Delay Measurements (ms)						
	1	2	3	4	5	6	Variance
1	75.997	74.871	74.710	74.553	74.493	73.987	0.451
2	71.778	71.757	71.372	71.232	71.215	70.949	0.107
3	69.802	69.643	69.251	69.027	68.976	68.586	0.203
4	68.485	67.558	67.564	67.390	67.213	66.805	0.311
5	65.776	65.560	65.591	65.354	65.398	65.379	0.027
6	64.921	64.795	64.591	64.666	64.381	64.383	0.047
7	46.856	47.312	47.519	47.681	47.833	47.987	0.165
8	64.280	64.127	64.243	64.113	63.976	63.487	0.084
9	62.995	63.146	62.453	62.487	62.523	62.610	0.086
10	62.568	62.403	62.034	62.266	62.129	61.857	0.066
11	61.483	61.364	60.999	60.775	60.654	60.307	0.197
12	47.563	47.784	48.004	47.525	48.346	48.523	0.169
13	59.973	59.845	59.758	59.446	59.478	59.487	0.050
14	59.293	59.027	58.929	58.862	58.872	58.511	0.065
15	57.703	57.809	57.399	57.175	57.005	57.162	0.104
16	57.701	57.612	57.524	57.304	57.268	57.198	0.042
17	57.091	56.922	56.845	56.718	56.678	56.327	0.068
18	55.920	55.666	55.531	55.419	55.403	55.139	0.070

Figure C.3: Cloud Platform 1 Delay

shown in Figure C.11. Mean values for Cloud Platform 2 are compared to the mean values of the baseline platform as well as other cloud platforms in Chapter IV.

C.4.2.3 Cloud Configuration 3.

Figures C.12, C.13, C.14 and C.15 show the measurements for packet loss, delay, jitter and throughput respectively under Cloud Platform 3, along with corresponding variances under all 18 traffic workload configurations. This data is used to construct the mean values



Cloud Platform 1 Jitter

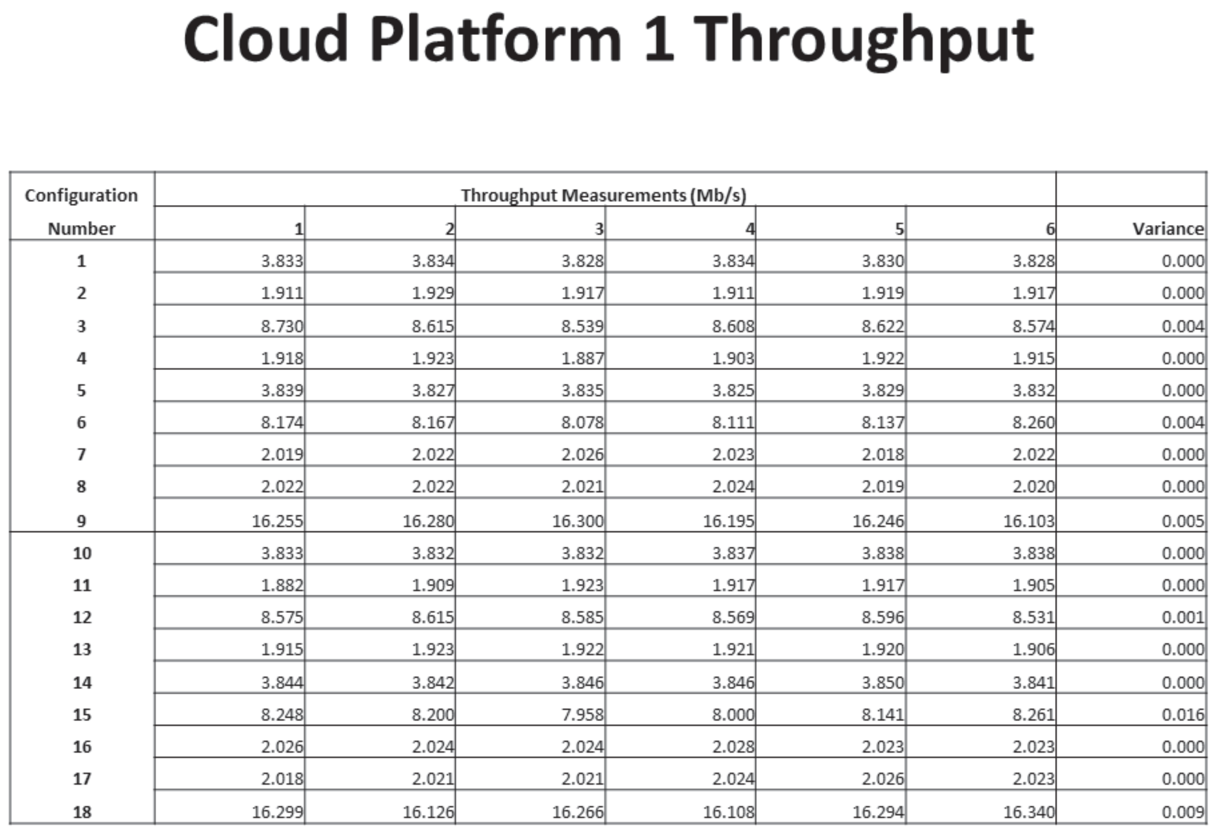
Configuration Number	Jitter Measurements (ms) 1	2	3	4	5	6	Variance
1	0.015	0.285	0.024	0.014	0.048	0.016	0.012
2	0.343	0.037	0.241	0.182	0.139	0.057	0.013
3	0.067	0.092	0.113	0.057	0.076	0.145	0.001
4	0.027	0.019	0.020	0.025	0.024	0.030	0.000
5	0.046	0.031	0.036	0.221	0.034	0.025	0.006
6	0.130	0.159	0.117	0.220	0.080	0.398	0.013
7	0.039	0.013	0.015	0.016	0.015	0.024	0.000
8	0.051	0.141	0.026	0.057	0.054	0.044	0.002
9	0.179	0.315	0.116	0.059	0.092	0.126	0.008
10	0.019	0.020	0.024	0.017	0.032	0.012	0.000
11	0.017	0.019	0.031	0.026	0.060	0.020	0.000
12	0.088	0.090	0.045	0.329	0.381	0.392	0.026
13	0.029	0.013	0.032	0.022	0.029	0.025	0.000
14	0.026	0.016	0.016	0.038	0.028	0.020	0.000
15	0.018	0.243	0.035	0.226	0.308	0.992	0.127
16	0.037	0.021	0.029	0.068	0.017	0.013	0.000
17	0.024	0.037	0.027	0.031	0.029	0.054	0.000
18	0.064	0.063	0.076	0.076	0.056	0.047	0.000

Figure C.4: Cloud Platform 1 Jitter

shown in Figure C.16. Mean values for Cloud Platform 3 are compared to the mean values of the baseline platform as well as other cloud platforms in Chapter IV.

C.4.2.4 Cloud Configuration 4.

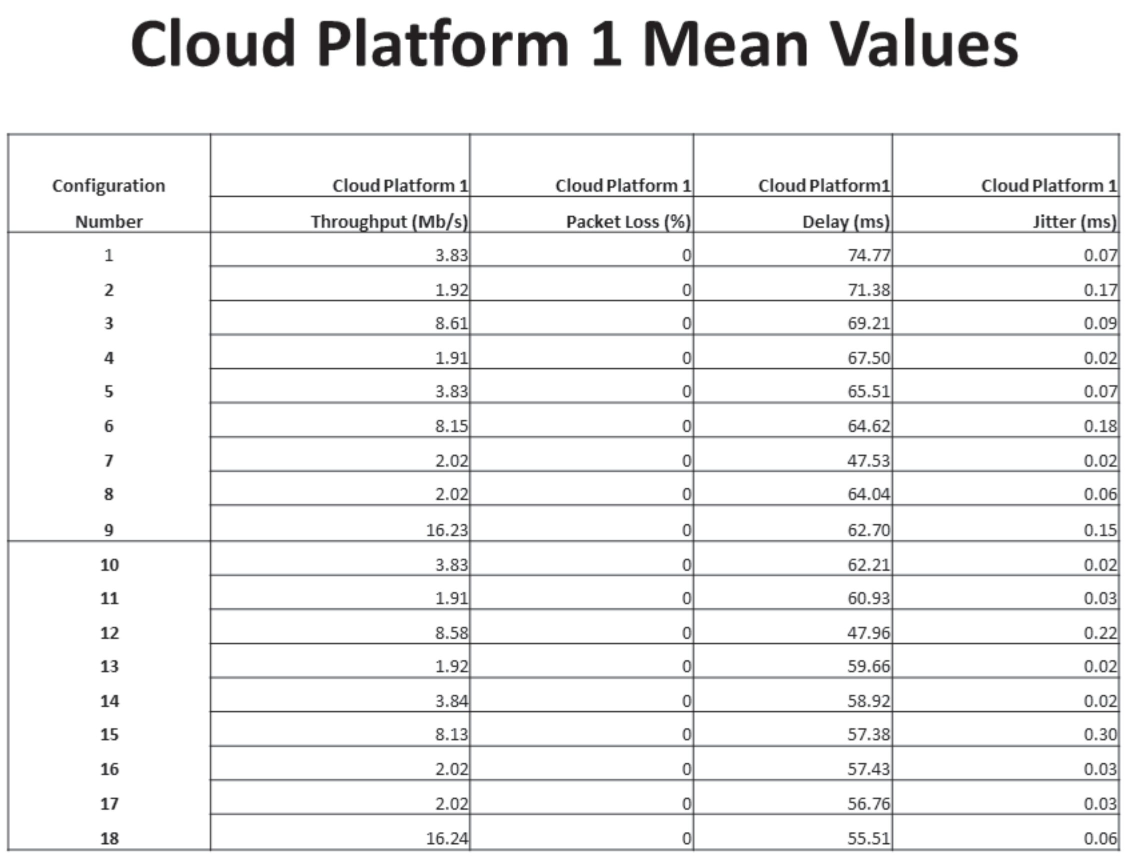
Figures C.17, C.18, C.19 and C.20 show the measurements for packet loss, delay, jitter and throughput respectively under Cloud Platform 4, along with corresponding variances under all 18 experimental configurations. This data is used to construct the mean values


Cloud Platform 1 Throughput

Configuration Number	Throughput Measurements (Mb/s)						
	1	2	3	4	5	6	Variance
1	3.833	3.834	3.828	3.834	3.830	3.828	0.000
2	1.911	1.929	1.917	1.911	1.919	1.917	0.000
3	8.730	8.615	8.539	8.608	8.622	8.574	0.004
4	1.918	1.923	1.887	1.903	1.922	1.915	0.000
5	3.839	3.827	3.835	3.825	3.829	3.832	0.000
6	8.174	8.167	8.078	8.111	8.137	8.260	0.004
7	2.019	2.022	2.026	2.023	2.018	2.022	0.000
8	2.022	2.022	2.021	2.024	2.019	2.020	0.000
9	16.255	16.280	16.300	16.195	16.246	16.103	0.005
10	3.833	3.832	3.832	3.837	3.838	3.838	0.000
11	1.882	1.909	1.923	1.917	1.917	1.905	0.000
12	8.575	8.615	8.585	8.569	8.596	8.531	0.001
13	1.915	1.923	1.922	1.921	1.920	1.906	0.000
14	3.844	3.842	3.846	3.846	3.850	3.841	0.000
15	8.248	8.200	7.958	8.000	8.141	8.261	0.016
16	2.026	2.024	2.024	2.028	2.023	2.023	0.000
17	2.018	2.021	2.021	2.024	2.026	2.023	0.000
18	16.299	16.126	16.266	16.108	16.294	16.340	0.009

Figure C.5: Cloud Platform 1 Throughput

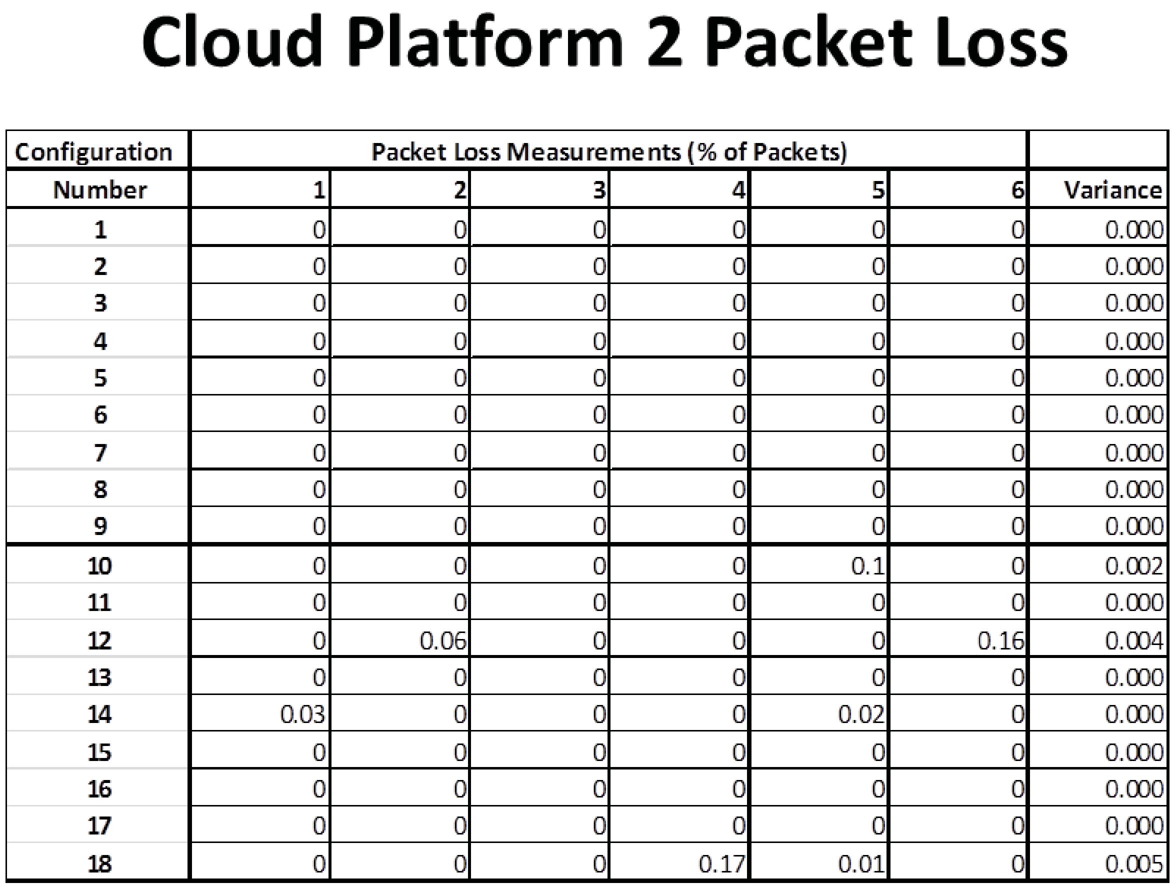
shown in Figure C.21. Mean values for Cloud Platform 4 are compared to the mean values of the baseline platform as well as other cloud platforms in Chapter IV.



Cloud Platform 1 Mean Values

Configuration Number	Cloud Platform 1 Throughput (Mb/s)	Cloud Platform 1 Packet Loss (%)	Cloud Platform1 Delay (ms)	Cloud Platform 1 Jitter (ms)
1	3.83	0	74.77	0.07
2	1.92	0	71.38	0.17
3	8.61	0	69.21	0.09
4	1.91	0	67.50	0.02
5	3.83	0	65.51	0.07
6	8.15	0	64.62	0.18
7	2.02	0	47.53	0.02
8	2.02	0	64.04	0.06
9	16.23	0	62.70	0.15
10	3.83	0	62.21	0.02
11	1.91	0	60.93	0.03
12	8.58	0	47.96	0.22
13	1.92	0	59.66	0.02
14	3.84	0	58.92	0.02
15	8.13	0	57.38	0.30
16	2.02	0	57.43	0.03
17	2.02	0	56.76	0.03
18	16.24	0	55.51	0.06

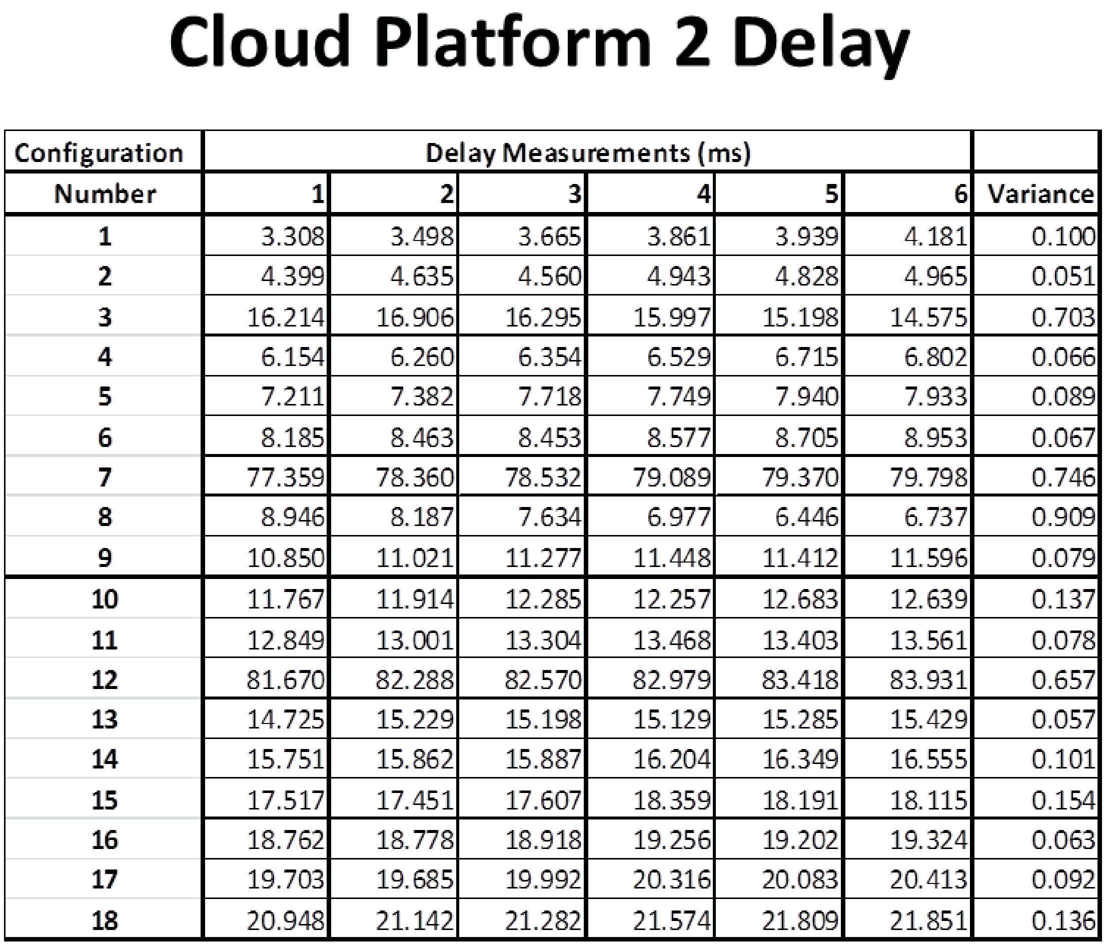
Figure C.6: Cloud Platform 1 Mean Values



Cloud Platform 2 Packet Loss

Configuration	Packet Loss Measurements (% of Packets)						
Number	1	2	3	4	5	6	Variance
1	0	0	0	0	0	0	0.000
2	0	0	0	0	0	0	0.000
3	0	0	0	0	0	0	0.000
4	0	0	0	0	0	0	0.000
5	0	0	0	0	0	0	0.000
6	0	0	0	0	0	0	0.000
7	0	0	0	0	0	0	0.000
8	0	0	0	0	0	0	0.000
9	0	0	0	0	0	0	0.000
10	0	0	0	0	0.1	0	0.002
11	0	0	0	0	0	0	0.000
12	0	0.06	0	0	0	0.16	0.004
13	0	0	0	0	0	0	0.000
14	0.03	0	0	0	0.02	0	0.000
15	0	0	0	0	0	0	0.000
16	0	0	0	0	0	0	0.000
17	0	0	0	0	0	0	0.000
18	0	0	0	0.17	0.01	0	0.005

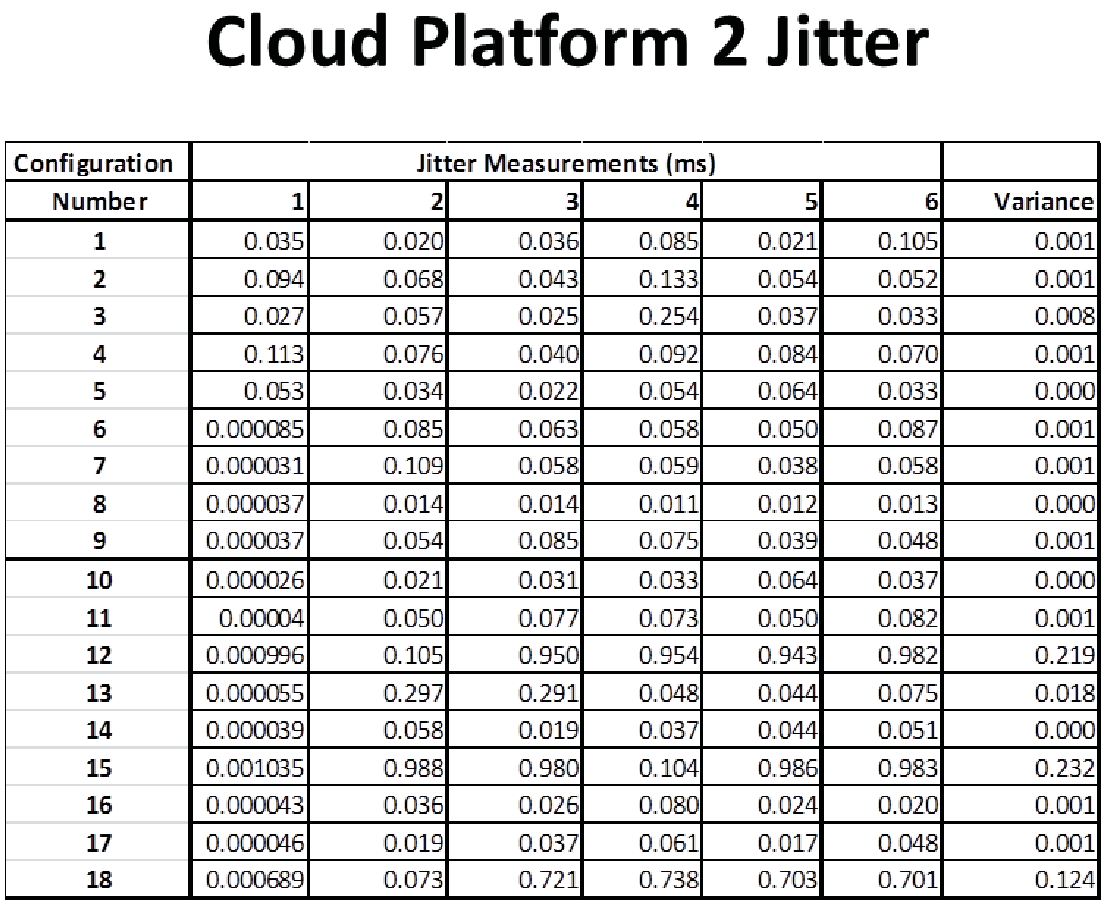
Figure C.7: Cloud Platform 2 Packet Loss



Cloud Platform 2 Delay

Configuration	Delay Measurements (ms)						
Number	1	2	3	4	5	6	Variance
1	3.308	3.498	3.665	3.861	3.939	4.181	0.100
2	4.399	4.635	4.560	4.943	4.828	4.965	0.051
3	16.214	16.906	16.295	15.997	15.198	14.575	0.703
4	6.154	6.260	6.354	6.529	6.715	6.802	0.066
5	7.211	7.382	7.718	7.749	7.940	7.933	0.089
6	8.185	8.463	8.453	8.577	8.705	8.953	0.067
7	77.359	78.360	78.532	79.089	79.370	79.798	0.746
8	8.946	8.187	7.634	6.977	6.446	6.737	0.909
9	10.850	11.021	11.277	11.448	11.412	11.596	0.079
10	11.767	11.914	12.285	12.257	12.683	12.639	0.137
11	12.849	13.001	13.304	13.468	13.403	13.561	0.078
12	81.670	82.288	82.570	82.979	83.418	83.931	0.657
13	14.725	15.229	15.198	15.129	15.285	15.429	0.057
14	15.751	15.862	15.887	16.204	16.349	16.555	0.101
15	17.517	17.451	17.607	18.359	18.191	18.115	0.154
16	18.762	18.778	18.918	19.256	19.202	19.324	0.063
17	19.703	19.685	19.992	20.316	20.083	20.413	0.092
18	20.948	21.142	21.282	21.574	21.809	21.851	0.136

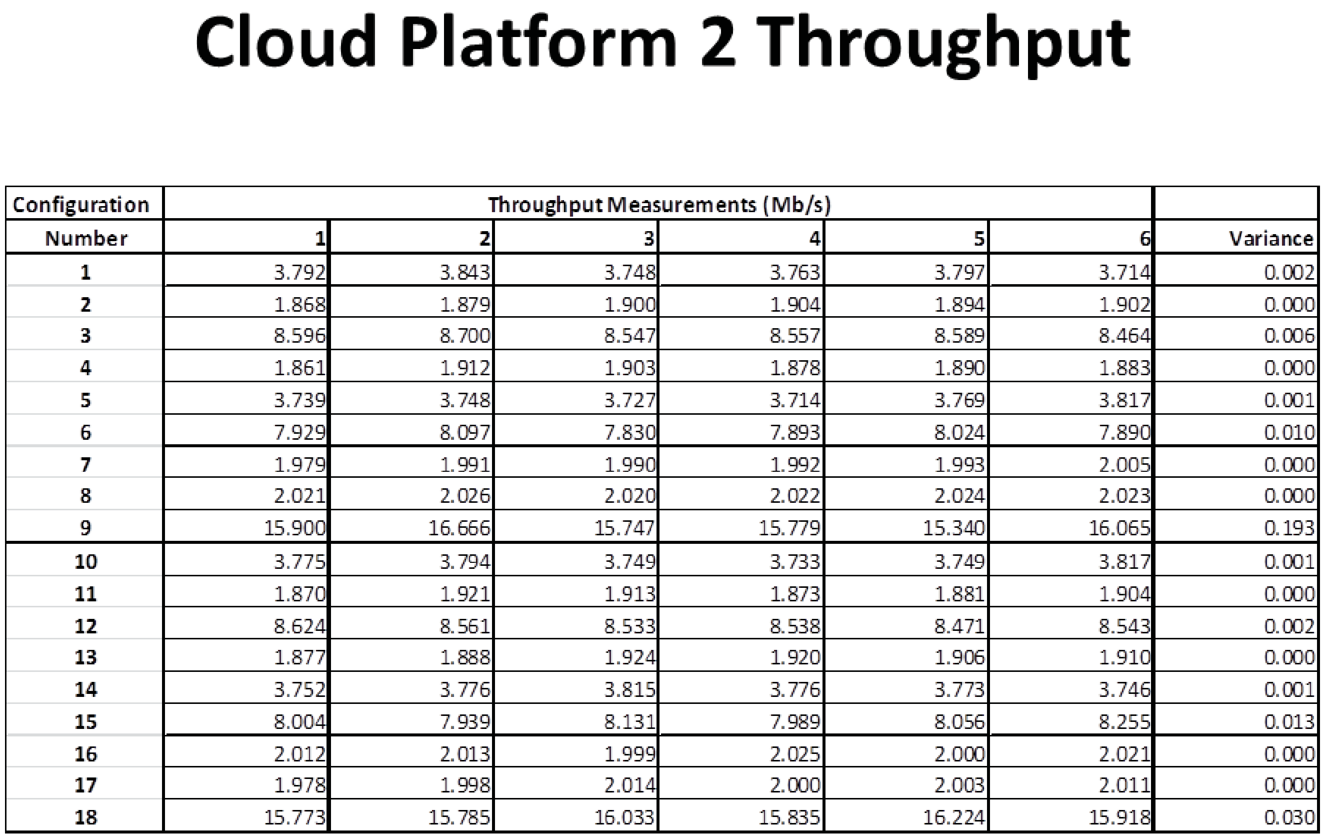
Figure C.8: Cloud Platform 2 Delay



Cloud Platform 2 Jitter

Configuration	Jitter Measurements (ms)						
Number	1	2	3	4	5	6	Variance
1	0.035	0.020	0.036	0.085	0.021	0.105	0.001
2	0.094	0.068	0.043	0.133	0.054	0.052	0.001
3	0.027	0.057	0.025	0.254	0.037	0.033	0.008
4	0.113	0.076	0.040	0.092	0.084	0.070	0.001
5	0.053	0.034	0.022	0.054	0.064	0.033	0.000
6	0.000085	0.085	0.063	0.058	0.050	0.087	0.001
7	0.000031	0.109	0.058	0.059	0.038	0.058	0.001
8	0.000037	0.014	0.014	0.011	0.012	0.013	0.000
9	0.000037	0.054	0.085	0.075	0.039	0.048	0.001
10	0.000026	0.021	0.031	0.033	0.064	0.037	0.000
11	0.00004	0.050	0.077	0.073	0.050	0.082	0.001
12	0.000996	0.105	0.950	0.954	0.943	0.982	0.219
13	0.000055	0.297	0.291	0.048	0.044	0.075	0.018
14	0.000039	0.058	0.019	0.037	0.044	0.051	0.000
15	0.001035	0.988	0.980	0.104	0.986	0.983	0.232
16	0.000043	0.036	0.026	0.080	0.024	0.020	0.001
17	0.000046	0.019	0.037	0.061	0.017	0.048	0.001
18	0.000689	0.073	0.721	0.738	0.703	0.701	0.124

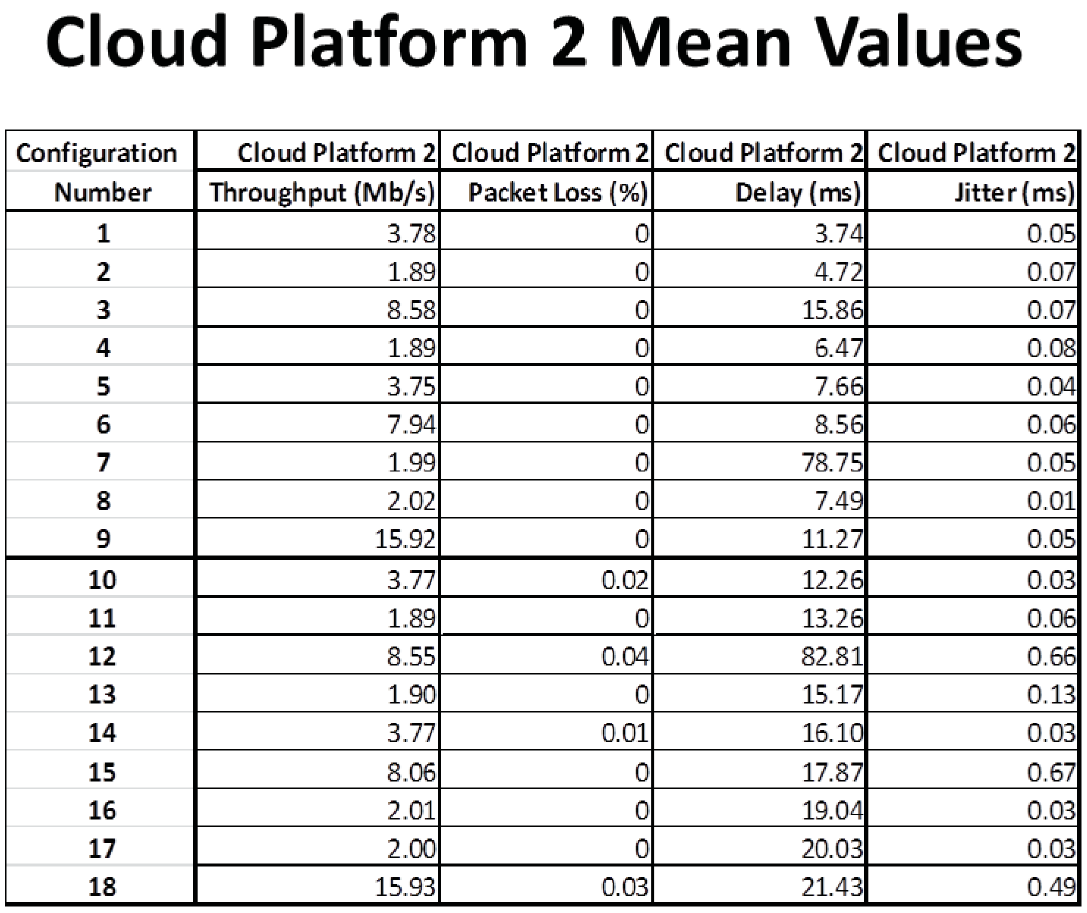
Figure C.9: Cloud Platform 2 Jitter



Cloud Platform 2 Throughput

Configuration	Throughput Measurements (Mb/s)						
Number	1	2	3	4	5	6	Variance
1	3.792	3.843	3.748	3.763	3.797	3.714	0.002
2	1.868	1.879	1.900	1.904	1.894	1.902	0.000
3	8.596	8.700	8.547	8.557	8.589	8.464	0.006
4	1.861	1.912	1.903	1.878	1.890	1.883	0.000
5	3.739	3.748	3.727	3.714	3.769	3.817	0.001
6	7.929	8.097	7.830	7.893	8.024	7.890	0.010
7	1.979	1.991	1.990	1.992	1.993	2.005	0.000
8	2.021	2.026	2.020	2.022	2.024	2.023	0.000
9	15.900	16.666	15.747	15.779	15.340	16.065	0.193
10	3.775	3.794	3.749	3.733	3.749	3.817	0.001
11	1.870	1.921	1.913	1.873	1.881	1.904	0.000
12	8.624	8.561	8.533	8.538	8.471	8.543	0.002
13	1.877	1.888	1.924	1.920	1.906	1.910	0.000
14	3.752	3.776	3.815	3.776	3.773	3.746	0.001
15	8.004	7.939	8.131	7.989	8.056	8.255	0.013
16	2.012	2.013	1.999	2.025	2.000	2.021	0.000
17	1.978	1.998	2.014	2.000	2.003	2.011	0.000
18	15.773	15.785	16.033	15.835	16.224	15.918	0.030

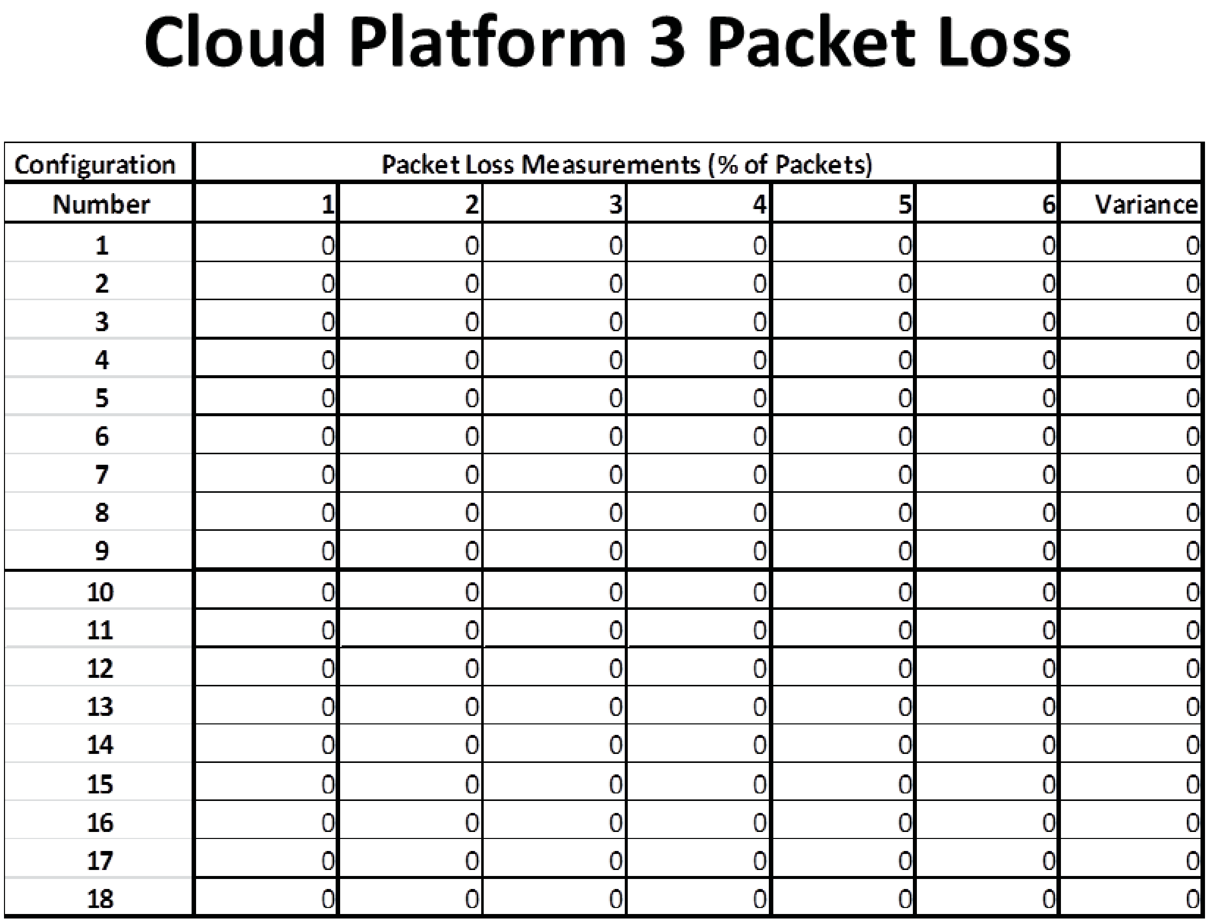
Figure C.10: Cloud Platform 2 Throughput



Cloud Platform 2 Mean Values

Configuration Number	Cloud Platform 2 Throughput (Mb/s)	Cloud Platform 2 Packet Loss (%)	Cloud Platform 2 Delay (ms)	Cloud Platform 2 Jitter (ms)
1	3.78	0	3.74	0.05
2	1.89	0	4.72	0.07
3	8.58	0	15.86	0.07
4	1.89	0	6.47	0.08
5	3.75	0	7.66	0.04
6	7.94	0	8.56	0.06
7	1.99	0	78.75	0.05
8	2.02	0	7.49	0.01
9	15.92	0	11.27	0.05
10	3.77	0.02	12.26	0.03
11	1.89	0	13.26	0.06
12	8.55	0.04	82.81	0.66
13	1.90	0	15.17	0.13
14	3.77	0.01	16.10	0.03
15	8.06	0	17.87	0.67
16	2.01	0	19.04	0.03
17	2.00	0	20.03	0.03
18	15.93	0.03	21.43	0.49

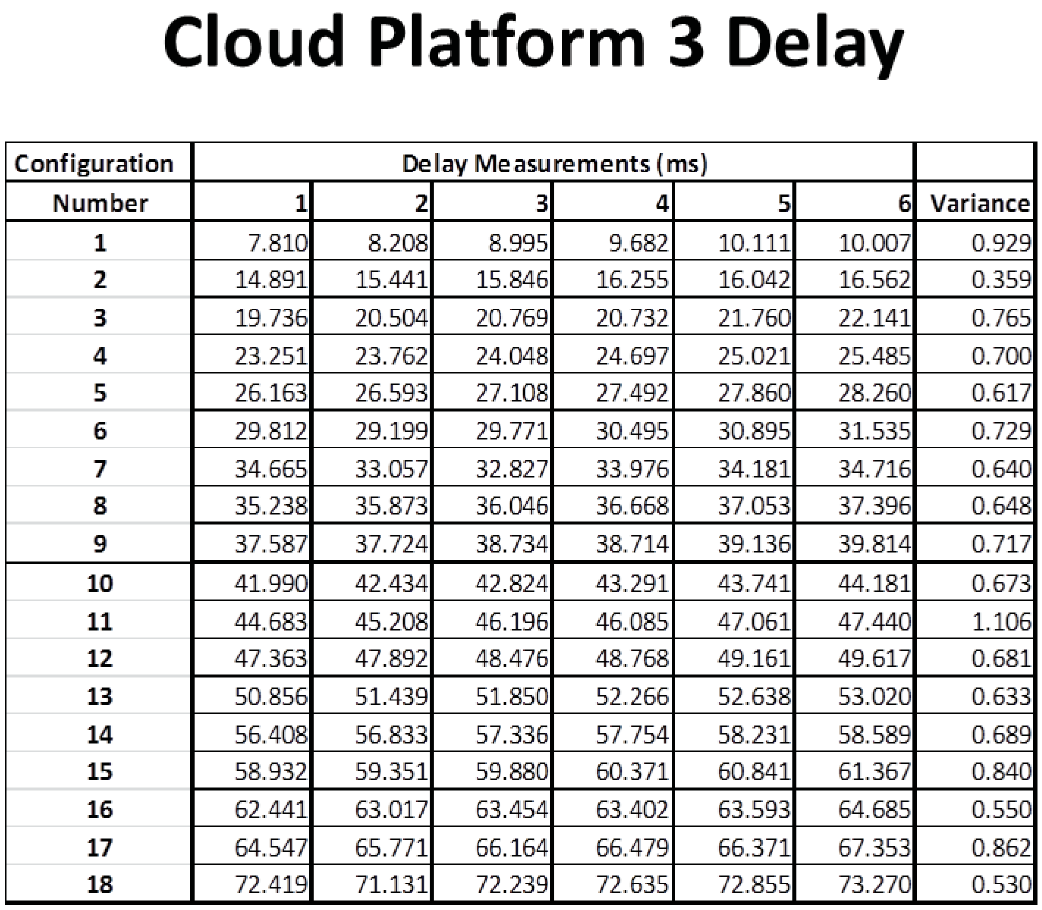
Figure C.11: Cloud Platform 2 Mean Values



Cloud Platform 3 Packet Loss

Configuration	Packet Loss Measurements (% of Packets)						
Number	1	2	3	4	5	6	Variance
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0

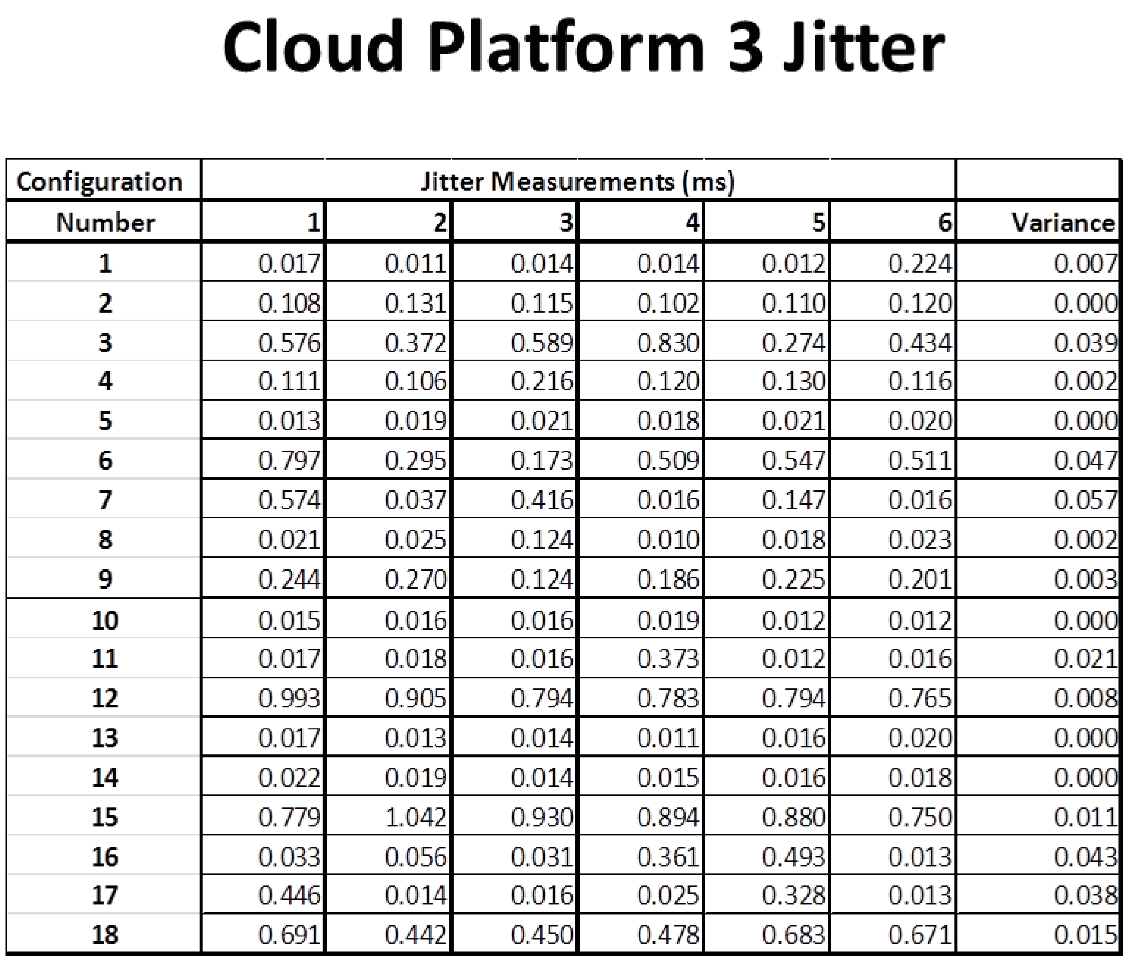
Figure C.12: Cloud Platform 3 Packet Loss



Cloud Platform 3 Delay

Configuration	Delay Measurements (ms)						
Number	1	2	3	4	5	6	Variance
1	7.810	8.208	8.995	9.682	10.111	10.007	0.929
2	14.891	15.441	15.846	16.255	16.042	16.562	0.359
3	19.736	20.504	20.769	20.732	21.760	22.141	0.765
4	23.251	23.762	24.048	24.697	25.021	25.485	0.700
5	26.163	26.593	27.108	27.492	27.860	28.260	0.617
6	29.812	29.199	29.771	30.495	30.895	31.535	0.729
7	34.665	33.057	32.827	33.976	34.181	34.716	0.640
8	35.238	35.873	36.046	36.668	37.053	37.396	0.648
9	37.587	37.724	38.734	38.714	39.136	39.814	0.717
10	41.990	42.434	42.824	43.291	43.741	44.181	0.673
11	44.683	45.208	46.196	46.085	47.061	47.440	1.106
12	47.363	47.892	48.476	48.768	49.161	49.617	0.681
13	50.856	51.439	51.850	52.266	52.638	53.020	0.633
14	56.408	56.833	57.336	57.754	58.231	58.589	0.689
15	58.932	59.351	59.880	60.371	60.841	61.367	0.840
16	62.441	63.017	63.454	63.402	63.593	64.685	0.550
17	64.547	65.771	66.164	66.479	66.371	67.353	0.862
18	72.419	71.131	72.239	72.635	72.855	73.270	0.530

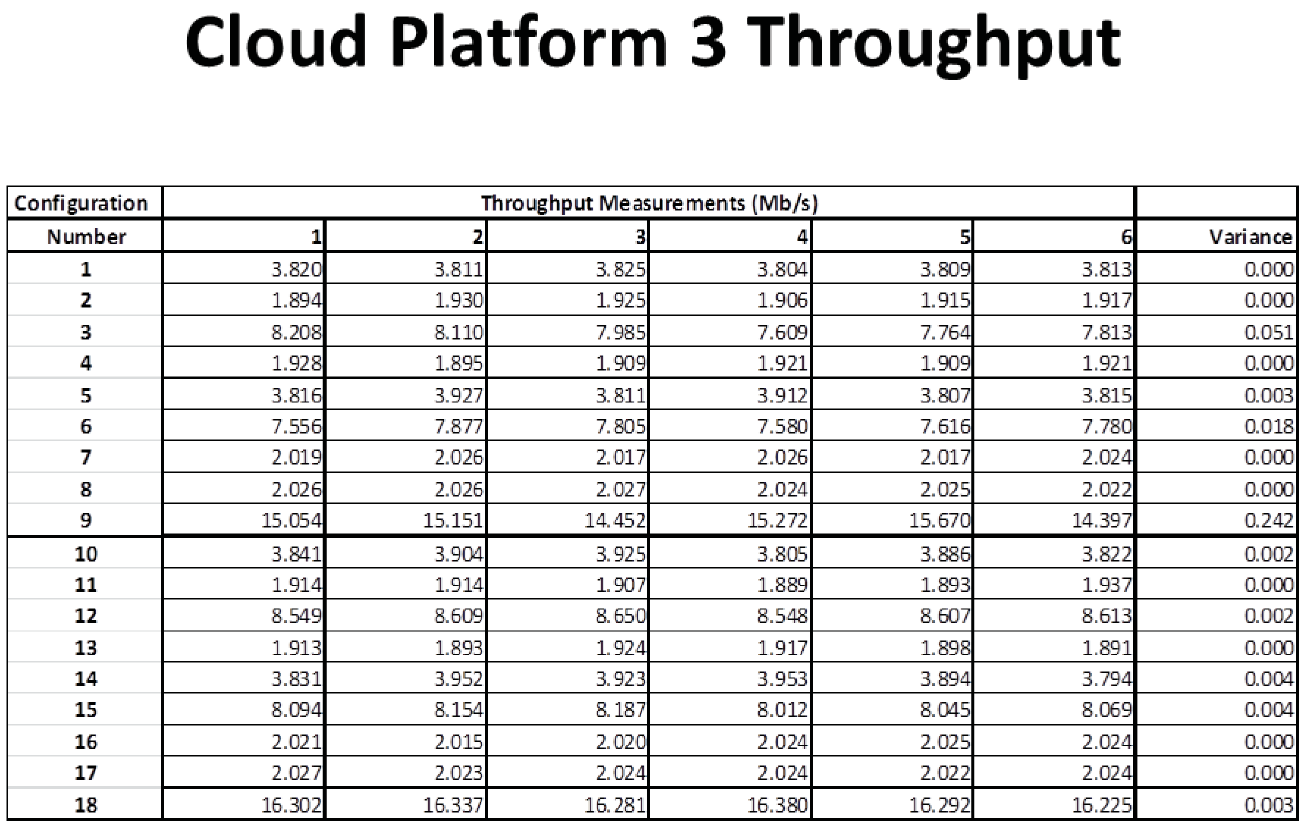
Figure C.13: Cloud Platform 3 Delay



Cloud Platform 3 Jitter

Configuration	Jitter Measurements (ms)						
Number	1	2	3	4	5	6	Variance
1	0.017	0.011	0.014	0.014	0.012	0.224	0.007
2	0.108	0.131	0.115	0.102	0.110	0.120	0.000
3	0.576	0.372	0.589	0.830	0.274	0.434	0.039
4	0.111	0.106	0.216	0.120	0.130	0.116	0.002
5	0.013	0.019	0.021	0.018	0.021	0.020	0.000
6	0.797	0.295	0.173	0.509	0.547	0.511	0.047
7	0.574	0.037	0.416	0.016	0.147	0.016	0.057
8	0.021	0.025	0.124	0.010	0.018	0.023	0.002
9	0.244	0.270	0.124	0.186	0.225	0.201	0.003
10	0.015	0.016	0.016	0.019	0.012	0.012	0.000
11	0.017	0.018	0.016	0.373	0.012	0.016	0.021
12	0.993	0.905	0.794	0.783	0.794	0.765	0.008
13	0.017	0.013	0.014	0.011	0.016	0.020	0.000
14	0.022	0.019	0.014	0.015	0.016	0.018	0.000
15	0.779	1.042	0.930	0.894	0.880	0.750	0.011
16	0.033	0.056	0.031	0.361	0.493	0.013	0.043
17	0.446	0.014	0.016	0.025	0.328	0.013	0.038
18	0.691	0.442	0.450	0.478	0.683	0.671	0.015

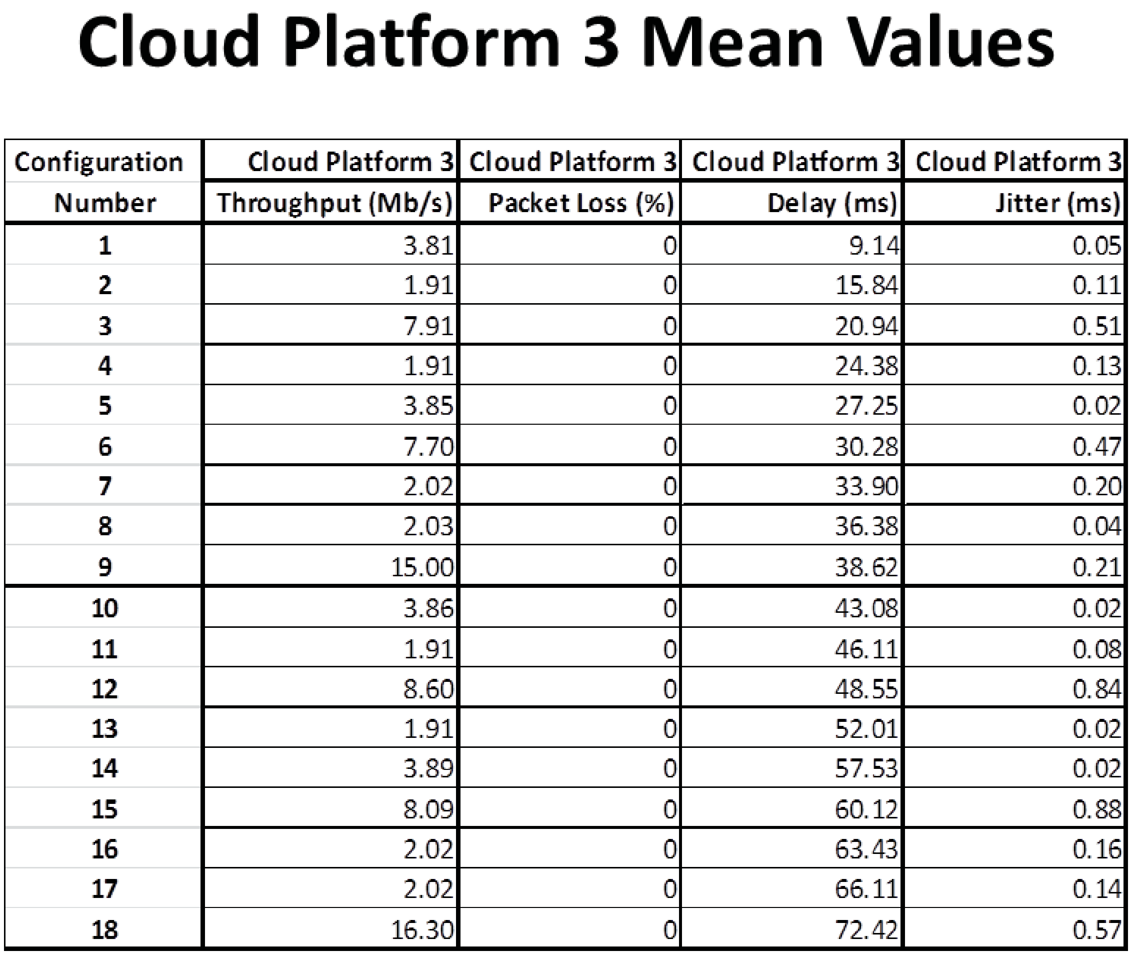
Figure C.14: Cloud Platform 3 Jitter



Cloud Platform 3 Throughput

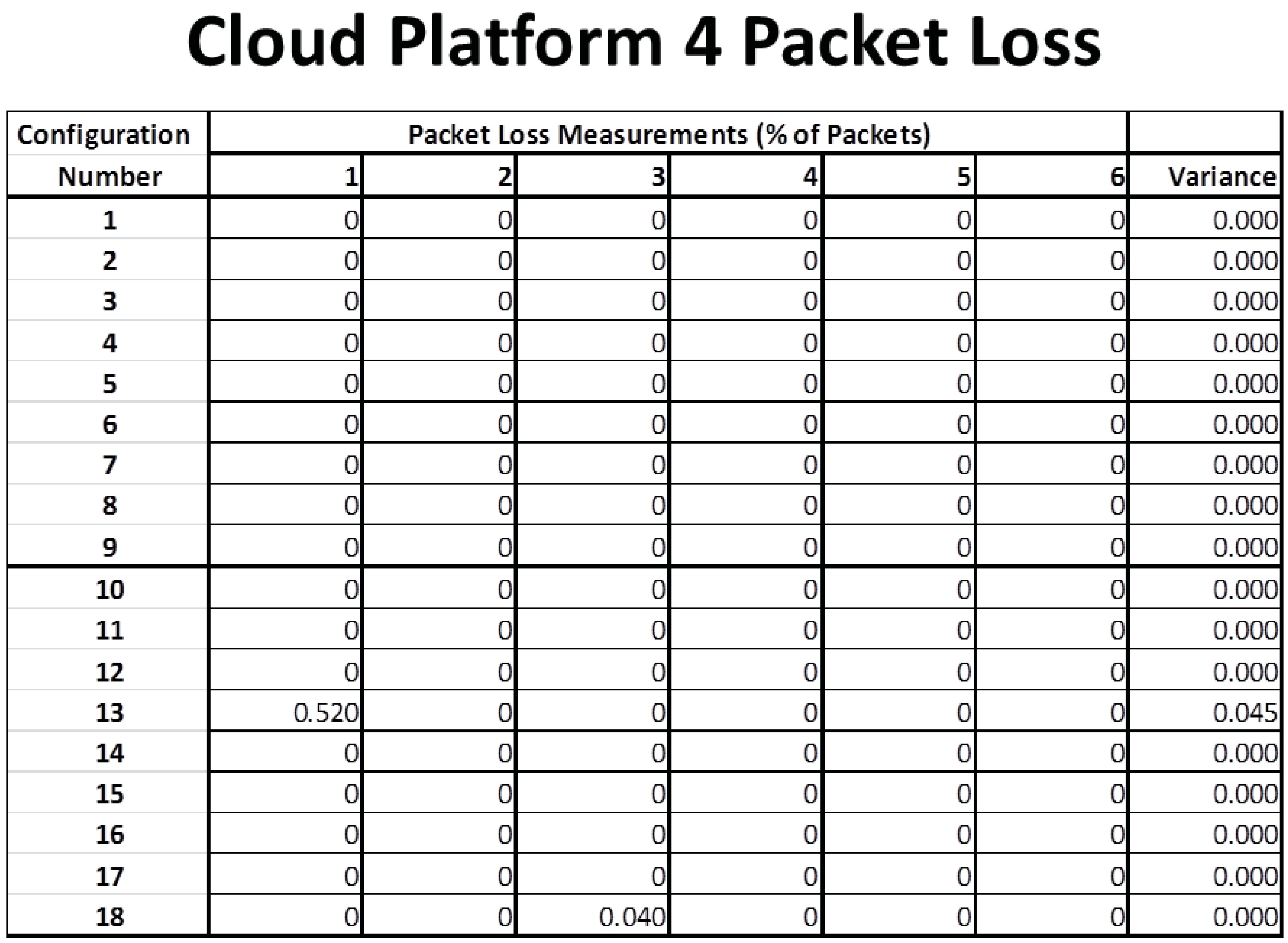
Configuration	Throughput Measurements (Mb/s)						
Number	1	2	3	4	5	6	Variance
1	3.820	3.811	3.825	3.804	3.809	3.813	0.000
2	1.894	1.930	1.925	1.906	1.915	1.917	0.000
3	8.208	8.110	7.985	7.609	7.764	7.813	0.051
4	1.928	1.895	1.909	1.921	1.909	1.921	0.000
5	3.816	3.927	3.811	3.912	3.807	3.815	0.003
6	7.556	7.877	7.805	7.580	7.616	7.780	0.018
7	2.019	2.026	2.017	2.026	2.017	2.024	0.000
8	2.026	2.026	2.027	2.024	2.025	2.022	0.000
9	15.054	15.151	14.452	15.272	15.670	14.397	0.242
10	3.841	3.904	3.925	3.805	3.886	3.822	0.002
11	1.914	1.914	1.907	1.889	1.893	1.937	0.000
12	8.549	8.609	8.650	8.548	8.607	8.613	0.002
13	1.913	1.893	1.924	1.917	1.898	1.891	0.000
14	3.831	3.952	3.923	3.953	3.894	3.794	0.004
15	8.094	8.154	8.187	8.012	8.045	8.069	0.004
16	2.021	2.015	2.020	2.024	2.025	2.024	0.000
17	2.027	2.023	2.024	2.024	2.022	2.024	0.000
18	16.302	16.337	16.281	16.380	16.292	16.225	0.003

Figure C.15: Cloud Platform 3 Throughput


Cloud Platform 3 Mean Values

Configuration	Cloud Platform 3	Cloud Platform 3	Cloud Platform 3	Cloud Platform 3
Number	Throughput (Mb/s)	Packet Loss (%)	Delay (ms)	Jitter (ms)
1	3.81	0	9.14	0.05
2	1.91	0	15.84	0.11
3	7.91	0	20.94	0.51
4	1.91	0	24.38	0.13
5	3.85	0	27.25	0.02
6	7.70	0	30.28	0.47
7	2.02	0	33.90	0.20
8	2.03	0	36.38	0.04
9	15.00	0	38.62	0.21
10	3.86	0	43.08	0.02
11	1.91	0	46.11	0.08
12	8.60	0	48.55	0.84
13	1.91	0	52.01	0.02
14	3.89	0	57.53	0.02
15	8.09	0	60.12	0.88
16	2.02	0	63.43	0.16
17	2.02	0	66.11	0.14
18	16.30	0	72.42	0.57

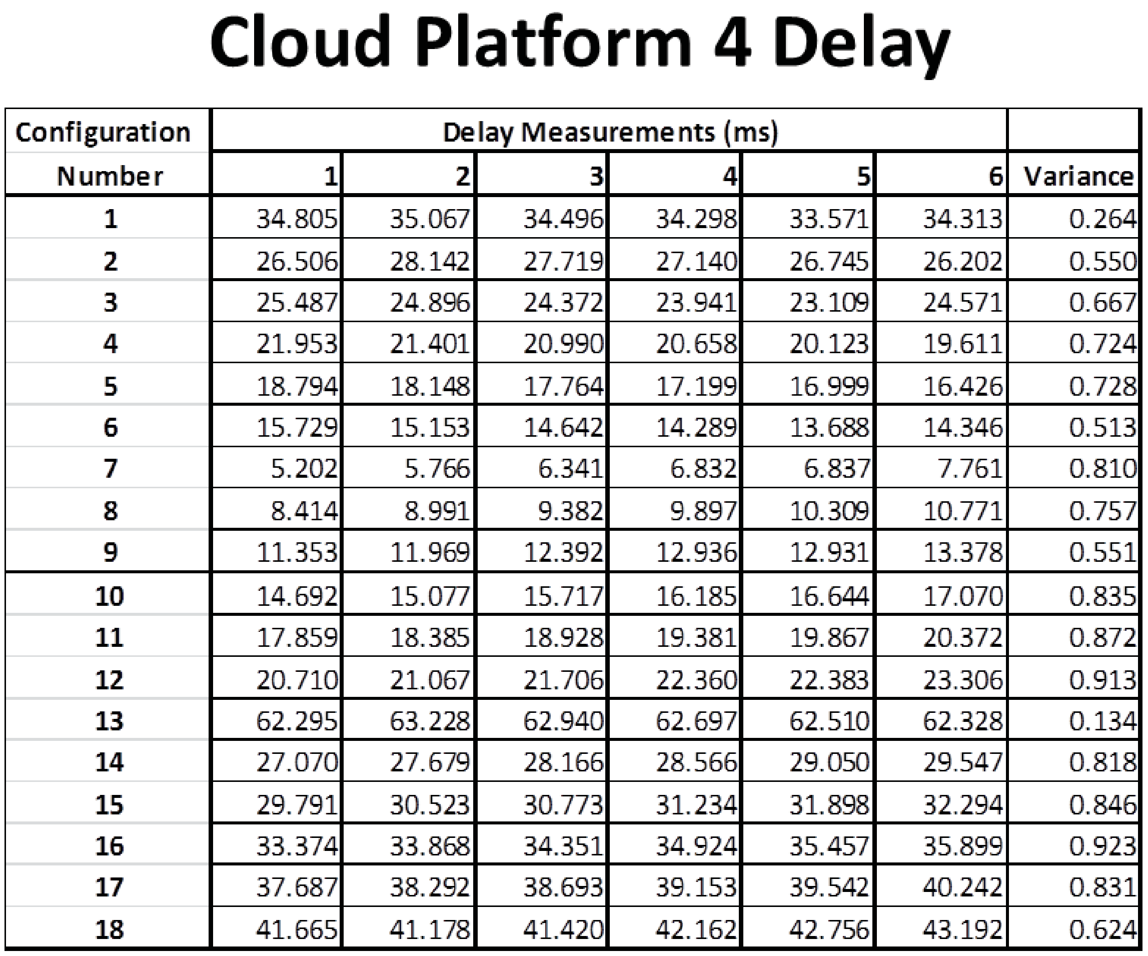
Figure C.16: Cloud Platform 3 Mean Values



Cloud Platform 4 Packet Loss

Configuration	Packet Loss Measurements (% of Packets)						
Number	1	2	3	4	5	6	Variance
1	0	0	0	0	0	0	0.000
2	0	0	0	0	0	0	0.000
3	0	0	0	0	0	0	0.000
4	0	0	0	0	0	0	0.000
5	0	0	0	0	0	0	0.000
6	0	0	0	0	0	0	0.000
7	0	0	0	0	0	0	0.000
8	0	0	0	0	0	0	0.000
9	0	0	0	0	0	0	0.000
10	0	0	0	0	0	0	0.000
11	0	0	0	0	0	0	0.000
12	0	0	0	0	0	0	0.000
13	0.520	0	0	0	0	0	0.045
14	0	0	0	0	0	0	0.000
15	0	0	0	0	0	0	0.000
16	0	0	0	0	0	0	0.000
17	0	0	0	0	0	0	0.000
18	0	0	0.040	0	0	0	0.000

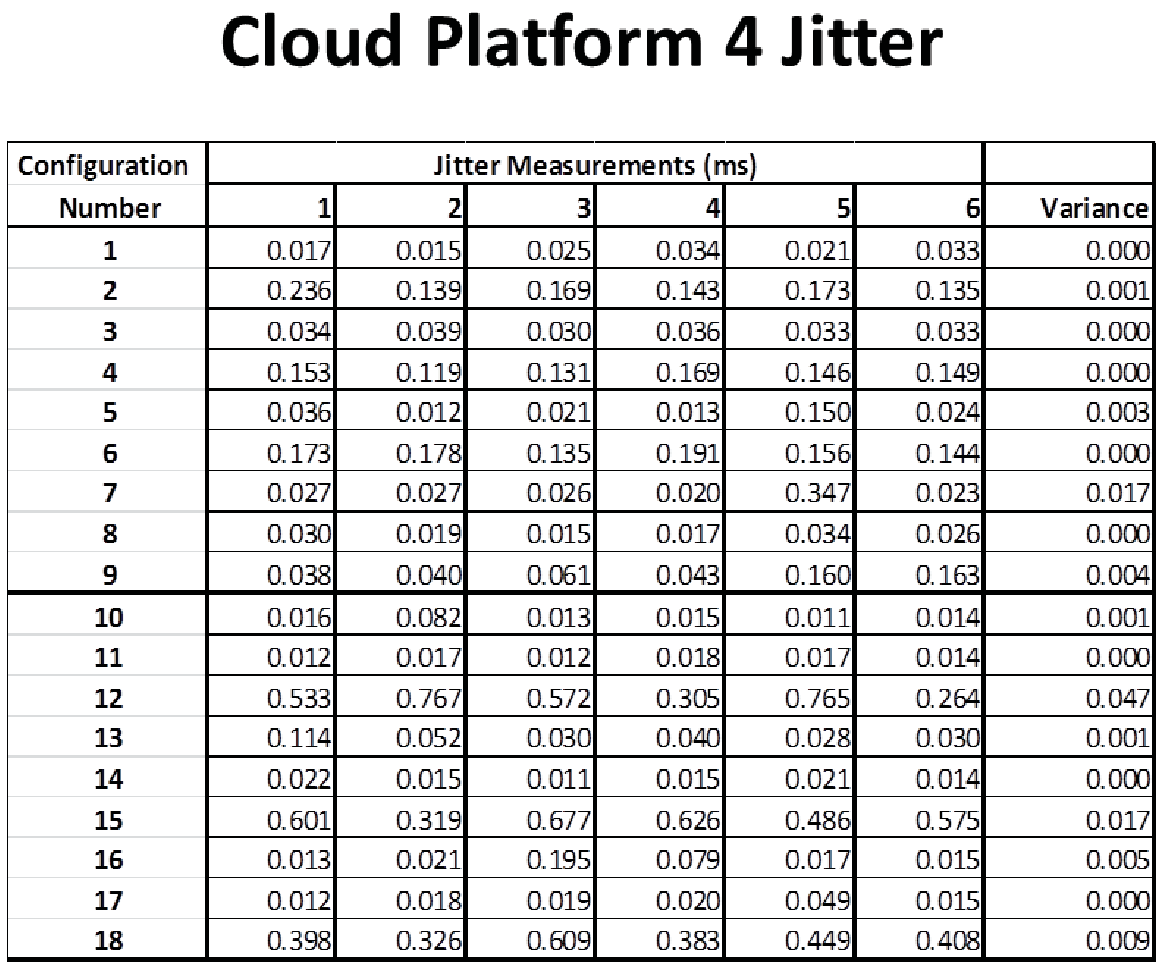
Figure C.17: Cloud Platform 4 Packet Loss



Cloud Platform 4 Delay

Configuration	Delay Measurements (ms)						
Number	1	2	3	4	5	6	Variance
1	34.805	35.067	34.496	34.298	33.571	34.313	0.264
2	26.506	28.142	27.719	27.140	26.745	26.202	0.550
3	25.487	24.896	24.372	23.941	23.109	24.571	0.667
4	21.953	21.401	20.990	20.658	20.123	19.611	0.724
5	18.794	18.148	17.764	17.199	16.999	16.426	0.728
6	15.729	15.153	14.642	14.289	13.688	14.346	0.513
7	5.202	5.766	6.341	6.832	6.837	7.761	0.810
8	8.414	8.991	9.382	9.897	10.309	10.771	0.757
9	11.353	11.969	12.392	12.936	12.931	13.378	0.551
10	14.692	15.077	15.717	16.185	16.644	17.070	0.835
11	17.859	18.385	18.928	19.381	19.867	20.372	0.872
12	20.710	21.067	21.706	22.360	22.383	23.306	0.913
13	62.295	63.228	62.940	62.697	62.510	62.328	0.134
14	27.070	27.679	28.166	28.566	29.050	29.547	0.818
15	29.791	30.523	30.773	31.234	31.898	32.294	0.846
16	33.374	33.868	34.351	34.924	35.457	35.899	0.923
17	37.687	38.292	38.693	39.153	39.542	40.242	0.831
18	41.665	41.178	41.420	42.162	42.756	43.192	0.624

Figure C.18: Cloud Platform 4 Delay



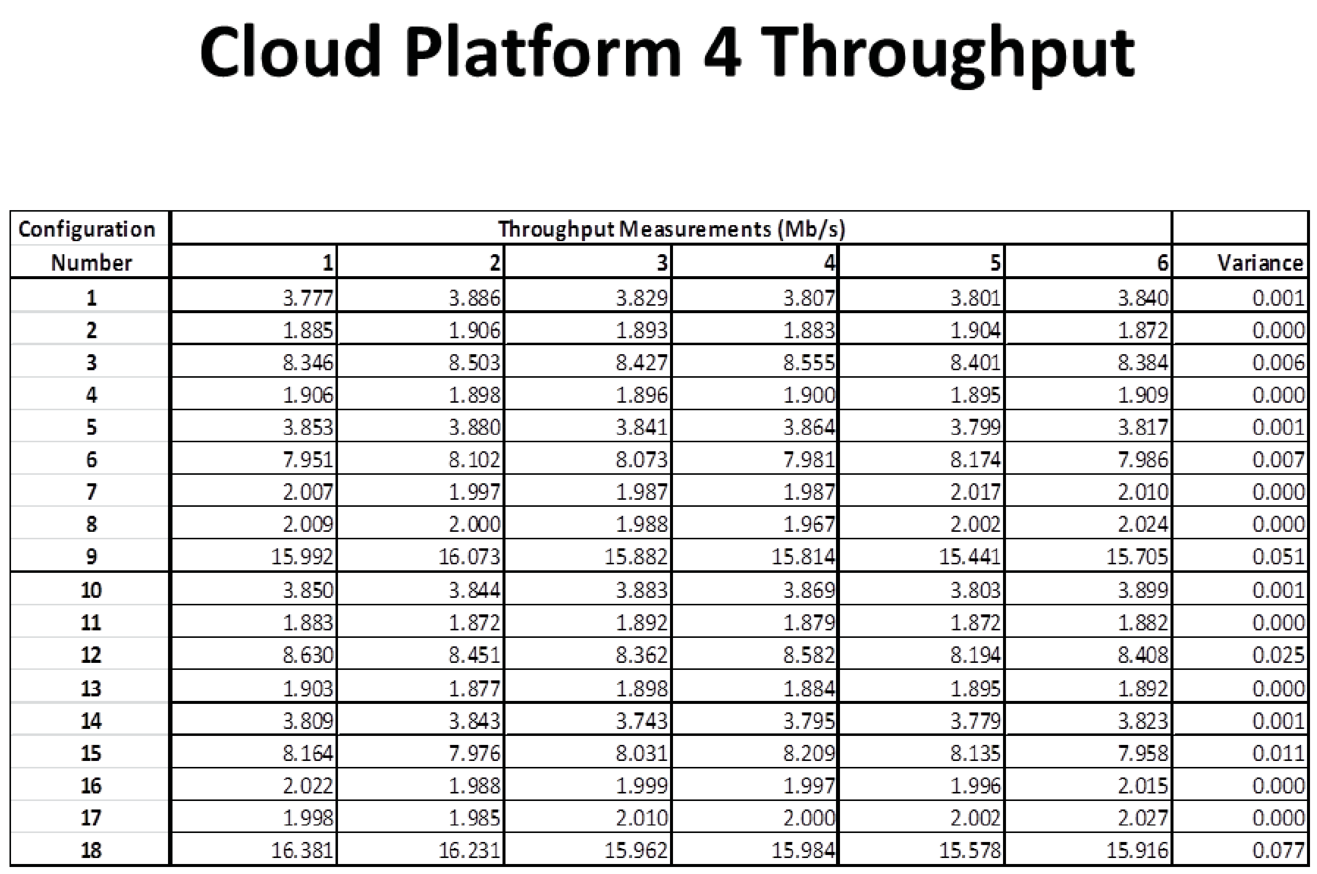
Cloud Platform 4 Jitter

Configuration	Jitter Measurements (ms)						
Number	1	2	3	4	5	6	Variance
1	0.017	0.015	0.025	0.034	0.021	0.033	0.000
2	0.236	0.139	0.169	0.143	0.173	0.135	0.001
3	0.034	0.039	0.030	0.036	0.033	0.033	0.000
4	0.153	0.119	0.131	0.169	0.146	0.149	0.000
5	0.036	0.012	0.021	0.013	0.150	0.024	0.003
6	0.173	0.178	0.135	0.191	0.156	0.144	0.000
7	0.027	0.027	0.026	0.020	0.347	0.023	0.017
8	0.030	0.019	0.015	0.017	0.034	0.026	0.000
9	0.038	0.040	0.061	0.043	0.160	0.163	0.004
10	0.016	0.082	0.013	0.015	0.011	0.014	0.001
11	0.012	0.017	0.012	0.018	0.017	0.014	0.000
12	0.533	0.767	0.572	0.305	0.765	0.264	0.047
13	0.114	0.052	0.030	0.040	0.028	0.030	0.001
14	0.022	0.015	0.011	0.015	0.021	0.014	0.000
15	0.601	0.319	0.677	0.626	0.486	0.575	0.017
16	0.013	0.021	0.195	0.079	0.017	0.015	0.005
17	0.012	0.018	0.019	0.020	0.049	0.015	0.000
18	0.398	0.326	0.609	0.383	0.449	0.408	0.009

Figure C.19: Cloud Platform 4 Jitter

C.5 Conclusion

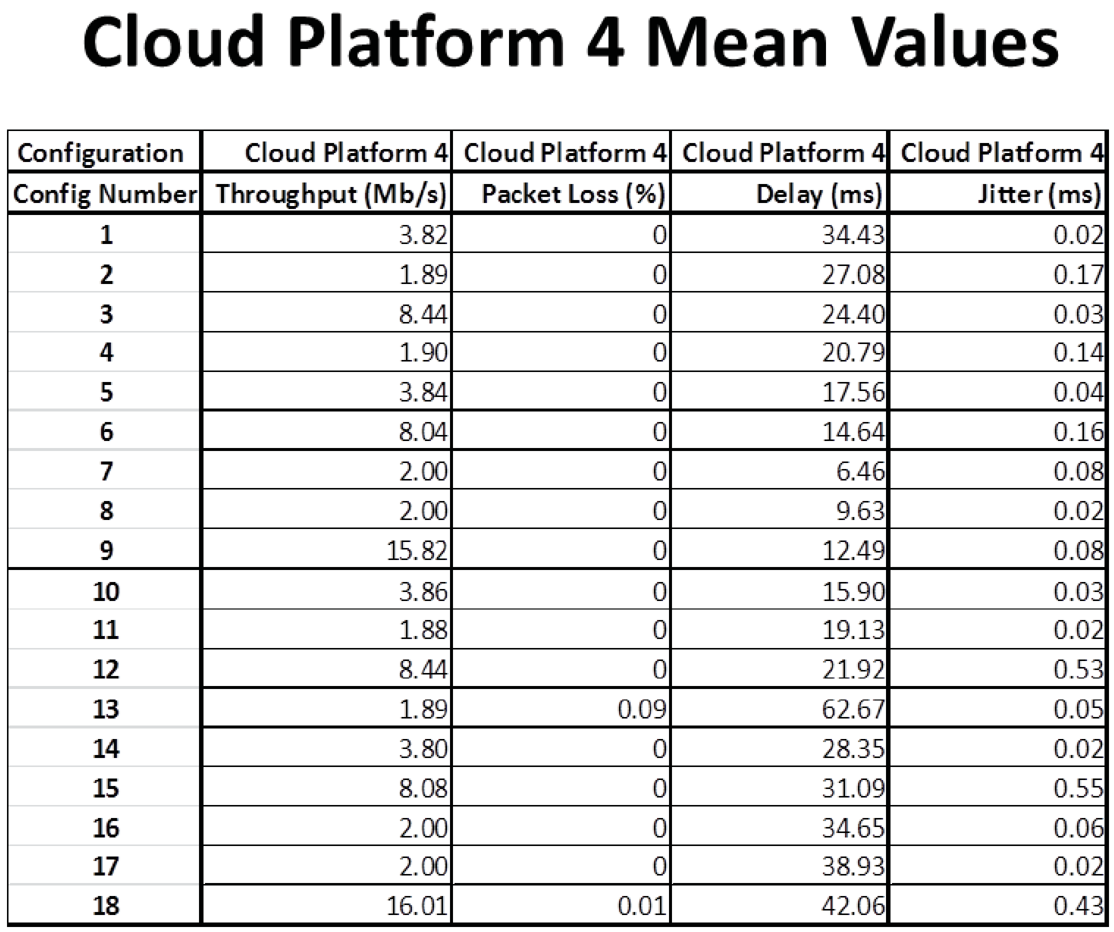
Experimental runs have sufficiently small variance to allow adequate comparison of mean values. These configurations represent typical traffic expected on a real network.



Cloud Platform 4 Throughput

Configuration	Throughput Measurements (Mb/s)						
Number	1	2	3	4	5	6	Variance
1	3.777	3.886	3.829	3.807	3.801	3.840	0.001
2	1.885	1.906	1.893	1.883	1.904	1.872	0.000
3	8.346	8.503	8.427	8.555	8.401	8.384	0.006
4	1.906	1.898	1.896	1.900	1.895	1.909	0.000
5	3.853	3.880	3.841	3.864	3.799	3.817	0.001
6	7.951	8.102	8.073	7.981	8.174	7.986	0.007
7	2.007	1.997	1.987	1.987	2.017	2.010	0.000
8	2.009	2.000	1.988	1.967	2.002	2.024	0.000
9	15.992	16.073	15.882	15.814	15.441	15.705	0.051
10	3.850	3.844	3.883	3.869	3.803	3.899	0.001
11	1.883	1.872	1.892	1.879	1.872	1.882	0.000
12	8.630	8.451	8.362	8.582	8.194	8.408	0.025
13	1.903	1.877	1.898	1.884	1.895	1.892	0.000
14	3.809	3.843	3.743	3.795	3.779	3.823	0.001
15	8.164	7.976	8.031	8.209	8.135	7.958	0.011
16	2.022	1.988	1.999	1.997	1.996	2.015	0.000
17	1.998	1.985	2.010	2.000	2.002	2.027	0.000
18	16.381	16.231	15.962	15.984	15.578	15.916	0.077

Figure C.20: Cloud Platform 4 Throughput



Cloud Platform 4 Mean Values

Configuration	Cloud Platform 4	Cloud Platform 4	Cloud Platform 4	Cloud Platform 4
Config Number	Throughput (Mb/s)	Packet Loss (%)	Delay (ms)	Jitter (ms)
1	3.82	0	34.43	0.02
2	1.89	0	27.08	0.17
3	8.44	0	24.40	0.03
4	1.90	0	20.79	0.14
5	3.84	0	17.56	0.04
6	8.04	0	14.64	0.16
7	2.00	0	6.46	0.08
8	2.00	0	9.63	0.02
9	15.82	0	12.49	0.08
10	3.86	0	15.90	0.03
11	1.88	0	19.13	0.02
12	8.44	0	21.92	0.53
13	1.89	0.09	62.67	0.05
14	3.80	0	28.35	0.02
15	8.08	0	31.09	0.55
16	2.00	0	34.65	0.06
17	2.00	0	38.93	0.02
18	16.01	0.01	42.06	0.43

Figure C.21: Cloud Platform 4 Mean Values

Bibliography

[1] Uoregon.edu. *Interpreting Test Statistics, p-values, and Significance*.

[2] Guruprasad, Shashikiran B. *Issues in Integrated Network Experimentation Using Simulation and Emulation*, Diss. The University of Utah edition, Aug 2005.

[3] Guruprasad, Sashi, Robert Ricci, and Jay Lepreau. *Integrated Network Experimentation using Simulation and Emulation*, Testbeds and Research Infrastructures for the Development of Networks and Communities, 2005. Tridentcom 2005. First International Conference on IEEE, 2005 edition, Feb 2005.

[4] Sadasivarao, Abhinava S. *Everest: Network Emulation Infrastructure in the Cloud*, MS thesis. Accessed from: http://www.contrib.andrew.cmu.edu/ asadasiv/everest/docs/everestedition, May 2012.

[5] Jianli Pan. *A Survey of Network Simulation Tools: Current Status and Future Developments*, Accessed from http://www.cse.wustl.edu/~jain/cse567-08/ftp/simtools/index.html#2 edition, Nov 2008.

[6] J. Heidemann et. al. *Effects of Detail in Wireless Network Simulation*, Proc. of the SCS Multiconference on Distributed Simulation, pp 3-11 edition, Jan 2001.

[7] K. Fall. *Network Emulation in the Vint/ns Simulator*, Proc. of the 4th IEEE Symposium on Computers and Communications edition, 1999.

[8] A. Vahdat et al. *Scalability and Accuracy in a Large-scale Network Emulator*, Proc. of the Fifth Symposium on Operating Systems Design and Implementation, pp 271-284, Boston, MA edition, Dec 2002.

[9] et al. B. White. *An Integrated Experimental Environment for Distributed Systems and Networks*, Proc. of the Fifth Symposium on Operating Systems Design and Implementation, pp. 255-270, Boston, MA edition, Dec 2002.

[10] *Network Simulator (ns-3)*, Accessed on Aug 4, 2013 from http://www.nsnam.org/ edition.

[11] O.T. Inc. *Network Modeling*, Accessed on Aug 4, 2013 from http://www.opnet.com/solutions/network_rd/modeler.html edition.

[12] Flux Research Group. *Emulab - Network Emulation Testbed Home*, Accessed on Aug 4, 2013 from http://www.emulab.net/ edition.

[13] *Emulab Experiments*, Accessed on Aug 4, 2013 from http:www.emulab.net/expubs.php edition.

[14] M.P. Kasick et. al. *Towards Fingerpointing in the Emulab Dynamic Distributed System*, Proceedings of the 3rd Conference on USENIX Workshop on Real, Large Distributed Systems, vol 3, pp 7, Berkeley, CA edition, 2006.

[15] B. Chun et. al. *Planetlab: An Overlay Testbed For Broad-coverage Services*, SIGCOMM Comput. Commun. Rev., vol 33, no. 3, pp 3-12 edition, Jul 2003.

[16] *Planetlab*, Accessed on Aug 4, 2013 from http://www.planet-lab.org/ edition.

[17] A. Bavier et. al. *In VINI Veritas: Realistic and Controlled Network Experimentation*, Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM, 06, New York, NY, pp 3-14 edition, 2006.

[18] O. Hodson M. Handley and E. Kohler. *Xorp: An Open Platform for Network Research*, SIGCOMM Comput. Commun. Rev, vol 33, no. 1, pp 53-57 edition, Jan 2003.

[19] et. al E. Kohler. *The Click Modular Router*, ACM Trans. Comput. Syst., vol. 18, no. 3, pp. 263-297 edition, Aug 2000.

[20] C. Francesco. *Amazon EC2*, Accessed on dec 10, 2013 from http://aws.amazon.com/ec2/ edition.

[21] Dipartimento di Informatica e Sistemistica Universita' degli Studi di Napoli "Federico II" (Italy). *D-ITG, Distributed Internet Traffic Generator*, Accessed on Dec 9, 2013 from http://traffic.comics.unina.it/software/ITG/ edition, Jul 2013.

[22] Alberto Dainotti Botta, Alessio and Antonio Pescapè. *A Tool for the Generation of Realistic Network Workload for Emerging Networking Scenarios*, Computer Networks, 2012, vol 56, num 15, pp 3531-3547. Accessed from http://dx.doi.org/10.1016/j.comnet.2012.02.019 edition, Mar 2012.

[23] R. Plackett and J. Burman. *The Design of Optimum Multifactorial Experiments*, Biometrika 33(4), pp 305-25 edition, Jun 1946.

[24] D. Roddy. *Satellite Communications (3 ed)*, Mcgraw-Hill. ISBN 0-07-137176-1 edition, 2001.

[25] R. Millar. *Response in Man-Machine Conversation Transactions*, Proc. AFIPS Fall Joint Computer Conference, vol 33, pp 267-277 edition, 2001.

[26] M. Claypool and K. Claypool. *Latency Can Kill: Precision and "Deadline in Online Games*, Accessed on Mar 1, 2014 from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.177.8852&rep=rep1&type=pdf edition, 2010.

[27] Openeaagles.org. *OpenEaagles Simulation Framework*, Accessed on Dec 20, 2013 from http://www.openeaagles.org/wiki/doku.php edition, 2001.

[28] P. Hintgens. *ZeroMQ - The Guide*, Accessed on Dec 10, 2013 from http://zguide.zeromq.org/page:all edition, Sep 2013.

[29] C. Francesco. *Europycon2011: Implementing Distributed Application using ZeroMQ*, Accessed on Dec 10, 2013 from http://www.slideshare.net/fcrippa/europycon2011-implementing-distributed-application-using-zeromq edition, Jul 2011.

[30] J. Ledolter and A. Swersey. *Testing 1-2-3: Experimental Design with Applications in Marketing and Service Operations*, Standford University Press, ISBN 978-0-8047-5612-9 edition, 2007.

[31] R. Bose and K. Kishen. *On the Problem of Confounding in the General Symmetrical Factorial Design*, Sankhya edition, May 1940.

[32] citrix.com. *Citrix and Amazon Web Services(AWS)*, Accessed on Dec 17, 2013 from http://www.citrix.com/global-partners/amazon-web-services.html edition.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704–0188*

1. REPORT DATE *(DD–MM–YYYY)* 27–03–2014	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Jun 2012-Mar 2014

4. TITLE AND SUBTITLE Leveraging The Cloud For Integrated Network Experimentation	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Beam, Brian A., Major, USA	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, Graduate School of Engineering and Management (AFIT/EN), 2950 Hobson Way, WPAFB, OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENG-14-M-11
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of the Secretary of Defense, Attn: Dr. Vincent Lillard, 1700 Defense Pentagon, Washington D.C. 20301, e-mail: vincent.lillard.ctr@osd.mil	**10. SPONSOR/MONITOR'S ACRONYM(S)** OSD
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES
This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT
The goal of this research is to determine the feasibility of performing integrated network experimentation using cloud services. This research uses performance metrics to compare computing architectures constructed in the cloud to architectures that run on traditional networks. If so, then cloud network architectures will display the same expected behavior as traditional network architectures, thus allowing the construction of networking testbeds at potentially substantial cost savings. Since the Amazon cloud does not support broadcast or multicast traffic, distributed applications face a challenge. Many distributed applications use broadcast or multicast to communicate real-time information. This research includes a case study for developing a distributed network application in the cloud which overcomes the restriction on broadcast and multicast traffic. During performance testing, the baseline network and cloud network configurations are provided statistically equivalent traffic workload. Metrics such as packet loss, delay, jitter and throughput are compared to determine relative performance. Analysis of the experimental results shows that in each case, the cloud network configurations performed at or above the performance level of the baseline network. Therefore, the public cloud infrastructure is suitable for performing integrated network experimentation. This research continues Project Everest's efforts to leverage cloud services for network experimentation. Project Everest is a framework which aims to combine emulation and cloud infrastructure into a single testbed using the Amazon Elastic Compute Cloud (EC2). Their tests indicate satisfactory cloud performance, but they recommend testing cloud network performance under various workload. This research carries out those performance tests.

15. SUBJECT TERMS
Simulation, Emulation, Cloud, Network Testbed, Integrated Network Experimentation

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dr. Douglas Hodson (AFIT/ENG)
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U	UU	122	(937) 255-3636 x4719 douglas.hodson@afit.edu